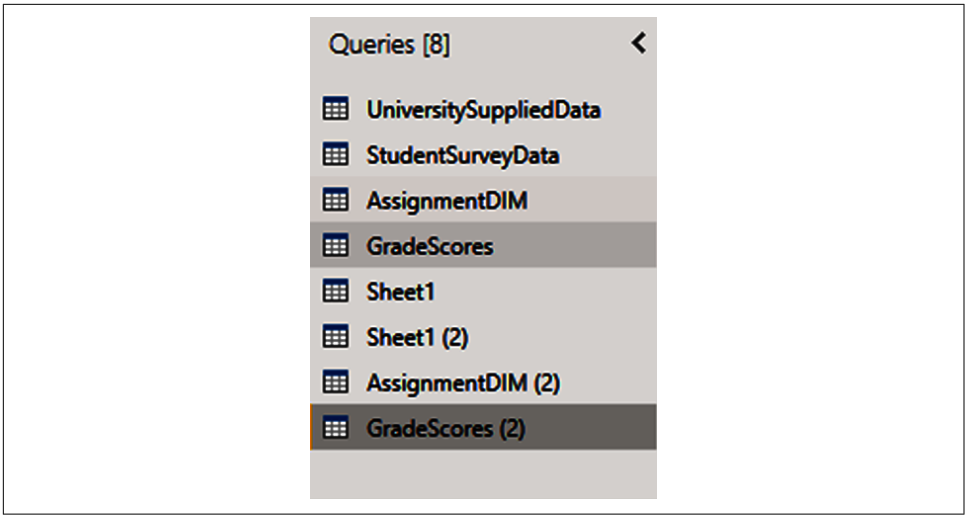


Figure 6-5. Your query list after importing your second set of data

Looking through these tables at a glance, it looks like AssignmentDIM is the same. There's no new data there, so let's remove that second AssignmentDIM from our queries list. How? You can either select it and press the Delete key or you can right-click it and select Delete.

Next let's rename those remaining sheets with more meaningful names. I did the university-provided data first and the survey data second, so my query list looks like Figure 6-6. Note that for my own clarity I added the (2) to the names in the list.

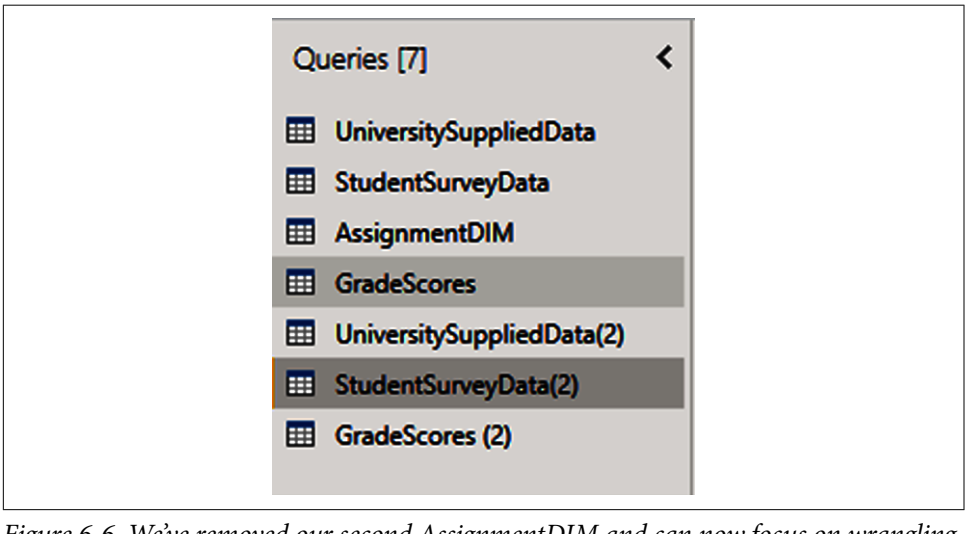


Figure 6-6. We've removed our second AssignmentDIM and can now focus on wrangling our three new queries

Consolidating Tables with Append

Inevitably, we don't want to have a bunch of extra tables in our final model, so we would like to consolidate these tables where we can. We can do that with the Append Queries function, as shown in the Combine section of the Home ribbon. If we choose the drop-down menu, we can select to Append or Append as New. In this case, we will just Append. Select the first table and click Append Queries. Select UniversitySuppliedData(2) from the "Table to append" list, and when we do that, we'll see what's shown in Figure 6-7.

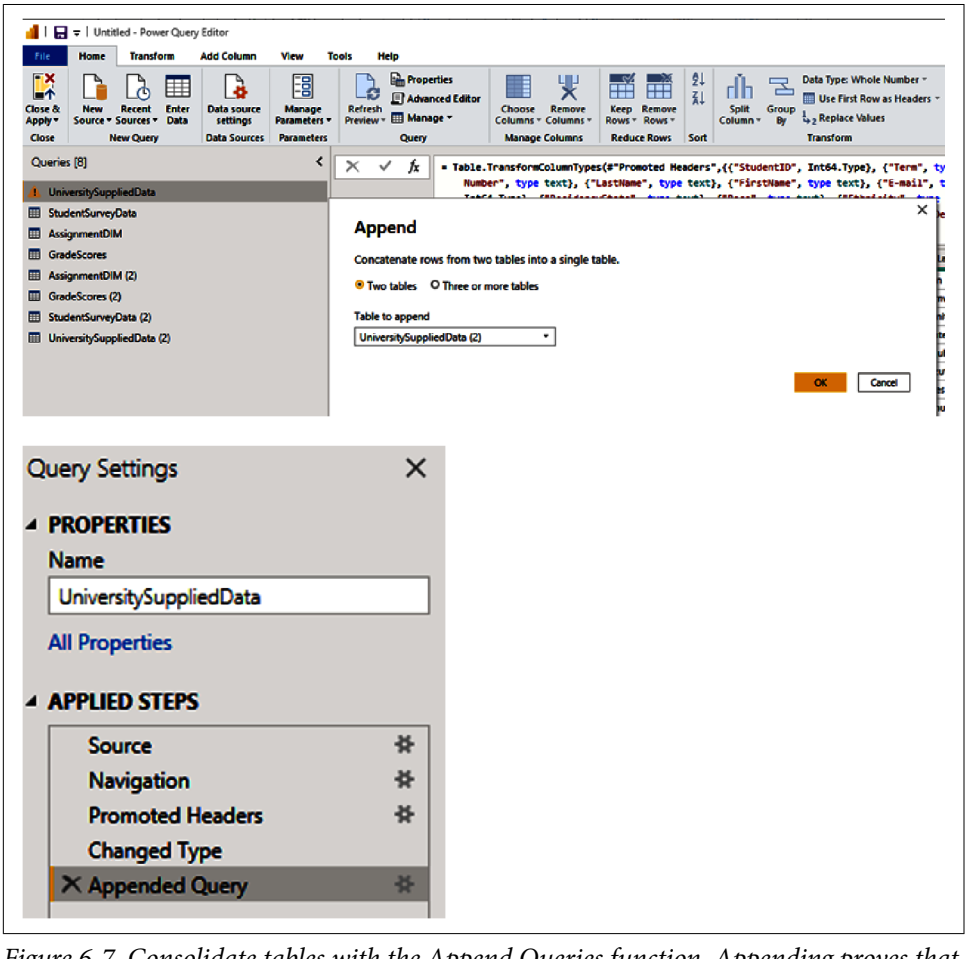


Figure 6-7. Consolidate tables with the Append Queries function. Appending proves that sometimes things are really better together.

Notice in the Applied Steps list that there is a new step called Appended Query, and we can see this single table now contains all the values from both UniversitySupplied-Data and UniversitySuppliedData(2). That's exactly what we want.

Now we cannot delete UniversitySuppliedData(2). That query is now referenced in the M script that allows the append to happen. When we finalize the load with the Close & Apply button, this table will also load. For demonstration purposes, I'm leaving them alone so that I can talk about hiding tables from the Report view later, but another option would be to right-click UniversitySuppliedData(2) and click "Enable load" so that the option is unchecked. This makes that query like a staging query that isn't loaded into the final dataset.

Now when we repeat the Append step for StudentSurveyData, we get a problem. Notice, I didn't say an error. There's nothing technically wrong. However, our data now has a problem. Look at the ClassID column in this appended StudentSurvey-Data. It's not unique. It has duplicated values, as we can see in Figure 6-8.

IDs should be unique. It's clear that the professor when considering these class IDs didn't think ahead to future courses and the impact of using the same IDs. However, we can create a unique ID for each record with something called an Index Column.

In the Add Column portion of the ribbon in the General section, you can see the Index Column with a drop-down. The default behavior for any Index Column is to start from 0, which really annoys me. So I will use the drop-down menu and select to start from 1. When we do that, we get a column named Index that appears at the far right side of the table.

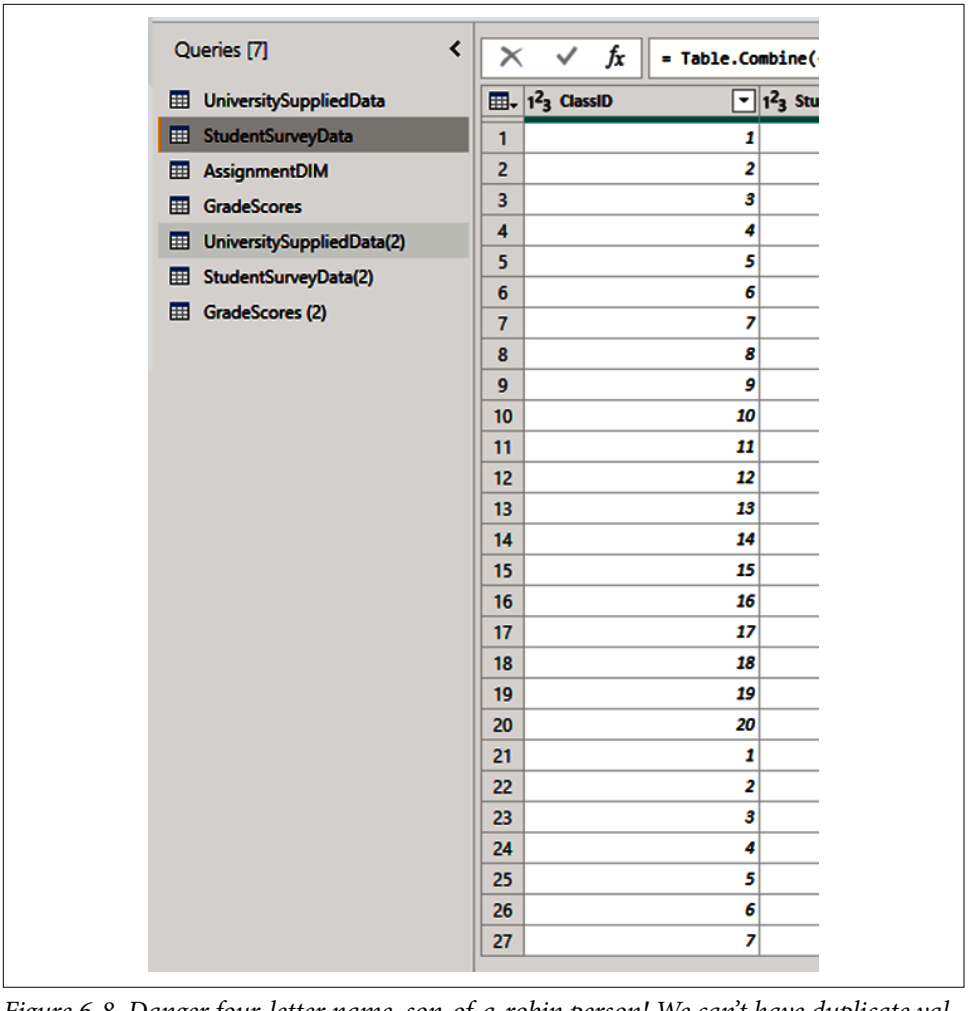


Figure 6-8. Danger four-letter name, son-of-a-robin person! We can't have duplicate values in a column that is supposed to be unique.

I personally like to see my primary ID on the left side, so I will right-click the column name and select Move → To Beginning to get it on the far left of the table. You could alternately click the Column Name, hold, and drag it all the way to the left. You can see the menu options when right-clicking a column in Figure 6-9.

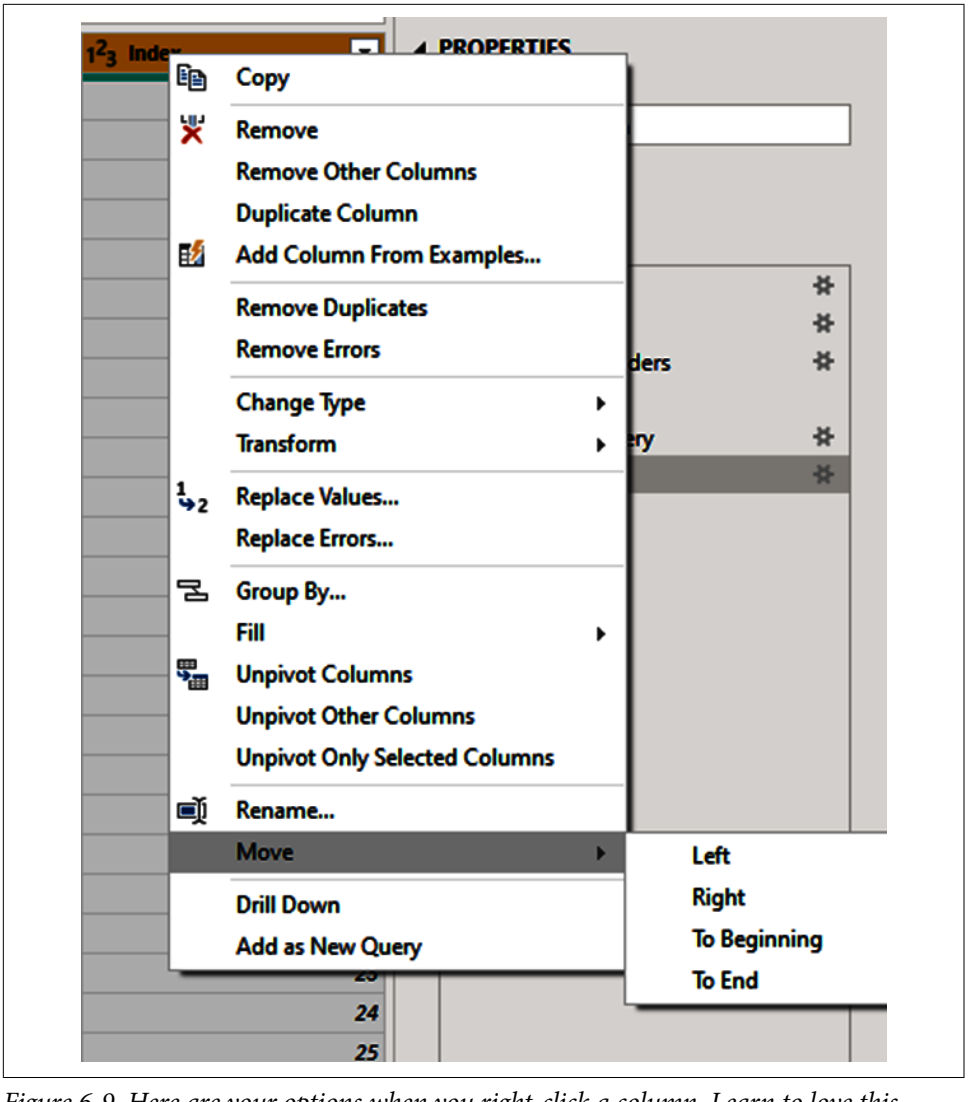


Figure 6-9. Here are your options when you right-click a column. Learn to love this menu, because you'll use it often.

We can also rename the Index Column, as we see the Rename option just above the Move option in Figure 6-9. In this case, I'm going to rename the Index to "TrueClassID" so that I know that's the unique value.

Using Merge to Get Columns from Other Tables

This gets our student survey data into a good spot, but our grades are going to be even more weird because of the issue with the ClassIDs. Notice that GradeScores and GradeScores (2) don't have any other way to identify students. If we were just to append them now, we really wouldn't have a good way to identify which person with ClassID 1 we're talking about. For instance, would it be the fall or summer student with that ID?

In this case, we need to find a unique value that we can match to each class ID. Thankfully, we have those in the student survey data tables in the StudentID. So we're going to do two merges here. We're going to merge GradeScores with StudentSurveyData to get the StudentID into that table, and then we will do the same thing with GradeScores (2) and StudentSurveyData(2).

Merging is a little more complicated than appending. Appending is, at its core, about making a table longer. Merging is about making a table wider. Another way to think about it is that appending is about adding rows, and merging is about adding columns.

Let's select GradeScores (2) and go back to the Home portion of the ribbon so we can select Merge Queries and perform the merge this time with StudentSurveyData(2). As with Append Queries, the drop-down menu will allow you to choose to Merge or Merge as New. We will again just merge. When you select Merge, you'll see an interface that is very reminiscent of the relationship creation interface. It autoselects the currently selected query and puts it on top, and then you can choose the second table, StudentSurveyData(2). Select ClassID in both tables. When we do that, we get what you see in Figure 6-10.

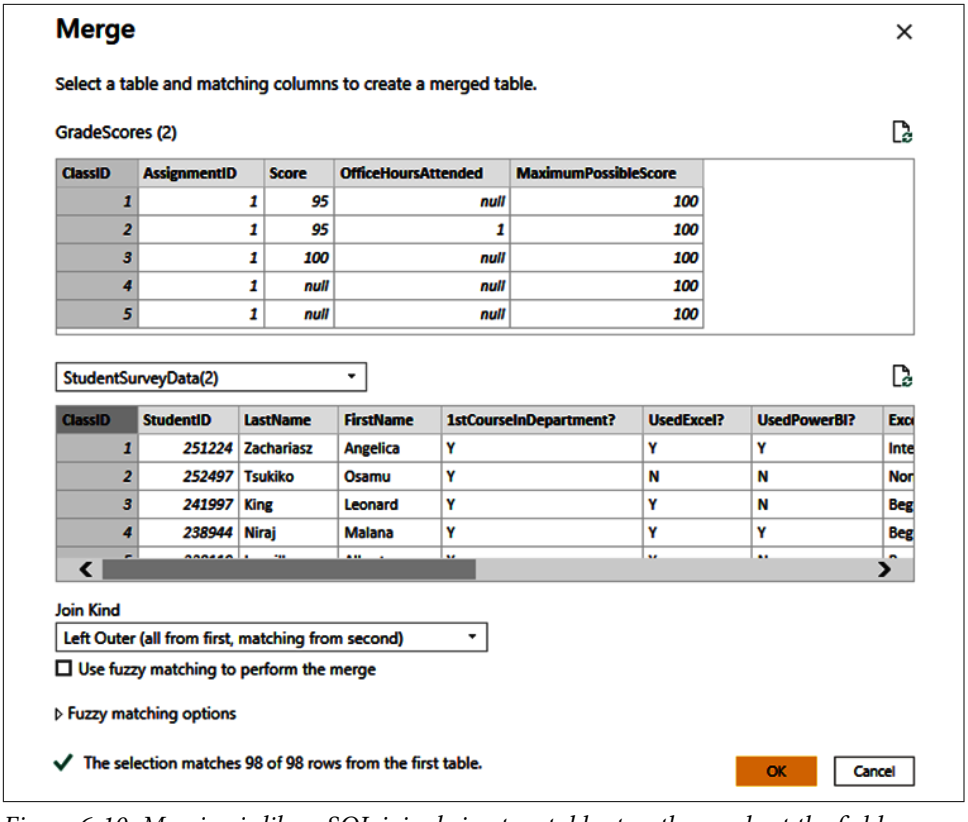


Figure 6-10. Merging is like a SQL join: bring two tables together and get the fields you need from each of them

Once the Merge is confirmed and you click OK, you'll notice that a column appears on the far right that has "Table" for every single record. This column also has a different symbol next to the column name, compared to the typical filter button on the other columns. That button next to the column name enables you to choose which columns you want to merge into the table. Figure 6-11 shows what that menu should look like. In this case, the only column I'm selecting is StudentID.

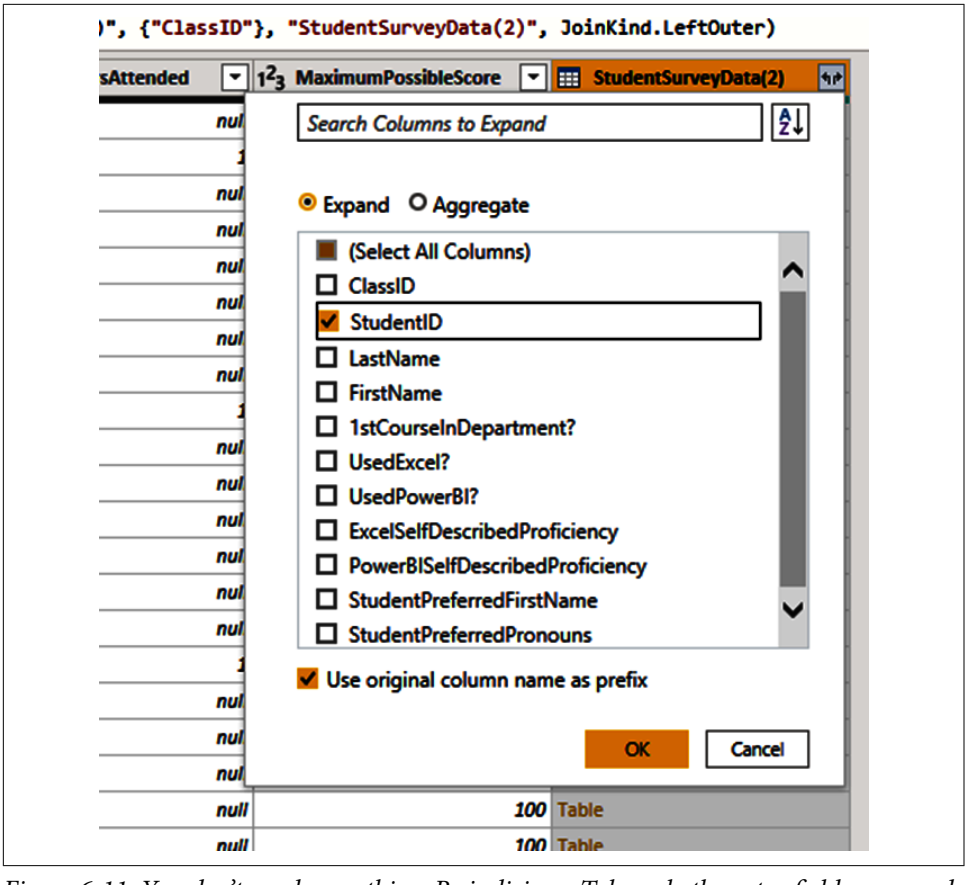
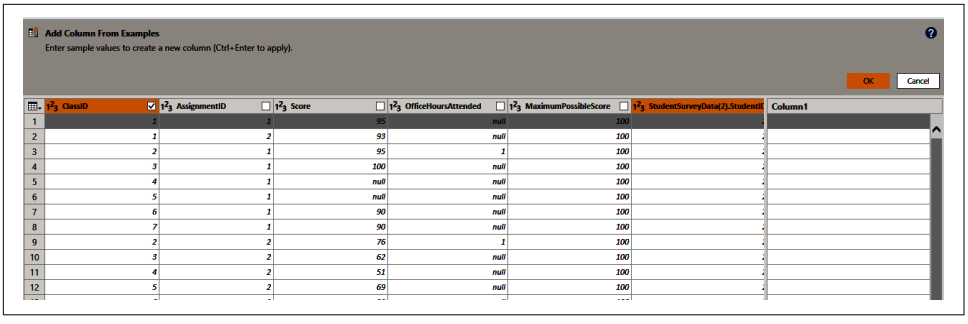


Figure 6-11. You don't need everything. Be judicious. Take only the extra fields you need.

Now that I have the StudentID column in this table, I can create a new column that will create a unique combination of ClassID and StudentID. We can do this by using the Column From Examples feature in the General section of the Add Column ribbon. To do this, hold down the Ctrl key and select the ClassID and StudentID columns so that we can choose From Selection from the button's drop-down list. Figure 6-12 shows the result.



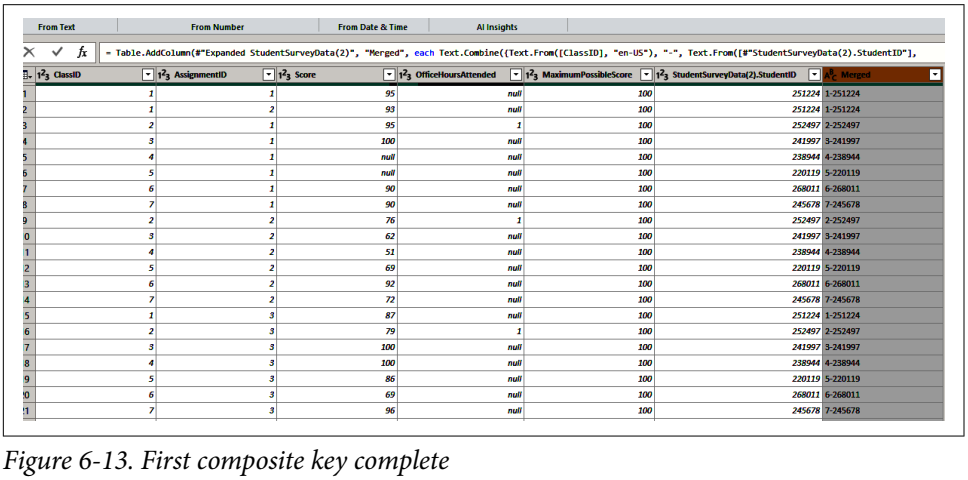
	ClassID	AssignmentID	Score	OfficeHoursAttended	MaximumPossibleScore	StudentSurveyData(2).StudentID	Column1
1	1	1	95	null	100		
2	1	2	93	null	100		
3	2	1	95	1	100		
4	3	1	100	null	100		
5	4	1	null	null	100		
6	5	1	null	null	100		
7	6	1	90	null	100		
8	7	1	90	null	100		
9	2	2	76	1	100		
10	3	2	62	null	100		
11	4	2	51	null	100		
12	5	2	69	null	100		

Figure 6-12. Column From Examples is an example of actual magic. Power Query's ability to intuit here is impressive.

Using this process, I'm going to create what I call a *composite key*. This will be a concatenated value or values that together form a unique combination for each record. In this case, I'll use ClassID and StudentID with a hyphen to separate them for readability, but this isn't required.

See Column 1 in Figure 6-12. In that first blank record, I put 1 for the listed ClassID, then a hyphen, and then the StudentID number, 251224. I get 1-251224. We can see that Column 1 will populate the appropriate values for all the other records, like an autofill process. You'll notice it's not perfect, and when I did this, it didn't quite get me the result I wanted.

I then went down the first record where the ClassID was 2 and put in 2-252497, and the rest of the records then filled the way I wanted. You can see that result in Figure 6-13. You'll notice in the gray area above the columns, it will also show you what M script is generated to populate the column. Click OK and then rename the column TrueClassStudentID.



	ClassID	AssignmentID	Score	OfficeHoursAttended	MaximumPossibleScore	StudentSurveyData(2).StudentID	Merged
1	1	1	95	null	100	251224	1-251224
2	1	2	93	null	100	251224	1-251224
3	2	1	95	1	100	252497	2-252497
4	3	1	100	null	100	241997	3-241997
5	4	1	null	null	100	238944	4-238944
6	5	1	null	null	100	220119	5-220119
7	6	1	90	null	100	268011	6-268011
8	7	1	90	null	100	245678	7-245678
9	2	2	76	1	100	252497	2-252497
0	3	2	62	null	100	241997	3-241997
1	4	2	51	null	100	238944	4-238944
2	5	2	69	null	100	220119	5-220119
3	6	2	92	null	100	268011	6-268011
4	7	2	72	null	100	245678	7-245678
5	1	3	87	null	100	251224	1-251224
6	2	3	79	1	100	252497	2-252497
7	3	3	100	null	100	241997	3-241997
8	4	3	100	null	100	238944	4-238944
9	5	3	86	null	100	220119	5-220119
0	6	3	69	null	100	268011	6-268011
1	7	3	96	null	100	245678	7-245678

Figure 6-13. First composite key complete

Next, we will add this TrueClassStudentID column to the StudentSurveyData table by creating a Column From Examples using ClassID and StudentID. It's the same thing we did with GradeScores (2) earlier. Remember that the StudentSurveyData table has already been appended at this point.

Now that we have our two composite keys, we can get the Student IDs to the original GradeScores table by using a merge on ClassID and TrueClassID. However, we're also going to use an inner join instead of a left outer join to keep the table clean. We will expand two columns into GradeScores: StudentID and TrueClassStudentID. When we have done these things, GradeScores should look like Figure 6-14.

With these steps done, we're just about ready to append GradeScores (2) to GradeScores. However, if we do it without making sure the column names are the same, we won't get the results we want, so make sure you rename the StudentID and TrueClassStudentID columns in both tables to ensure they're the same. Then you can append them together, as we did the tables before.

The last thing we could do now is to bring back in the TrueClassID from the StudentSurveyData into our GradeScores table via Merge. We do this so we'll have a nice, clean ID from 1 to 27 to work with merging on the TrueClassStudentID as an inner join condition. I'm going to go ahead and do that in my example.

Admittedly, I didn't do this in the most efficient way because I wanted to work through using Append and Merge, but a good exercise would be to take the steps we've talked about here and to optimize this transformation process to get all the data together.

At this point, our data has been transformed, and we're ready to load the data, build our relationships, and clean up our model. Click Close & Apply in the upper-left corner, and we'll move on to the next step of our work.

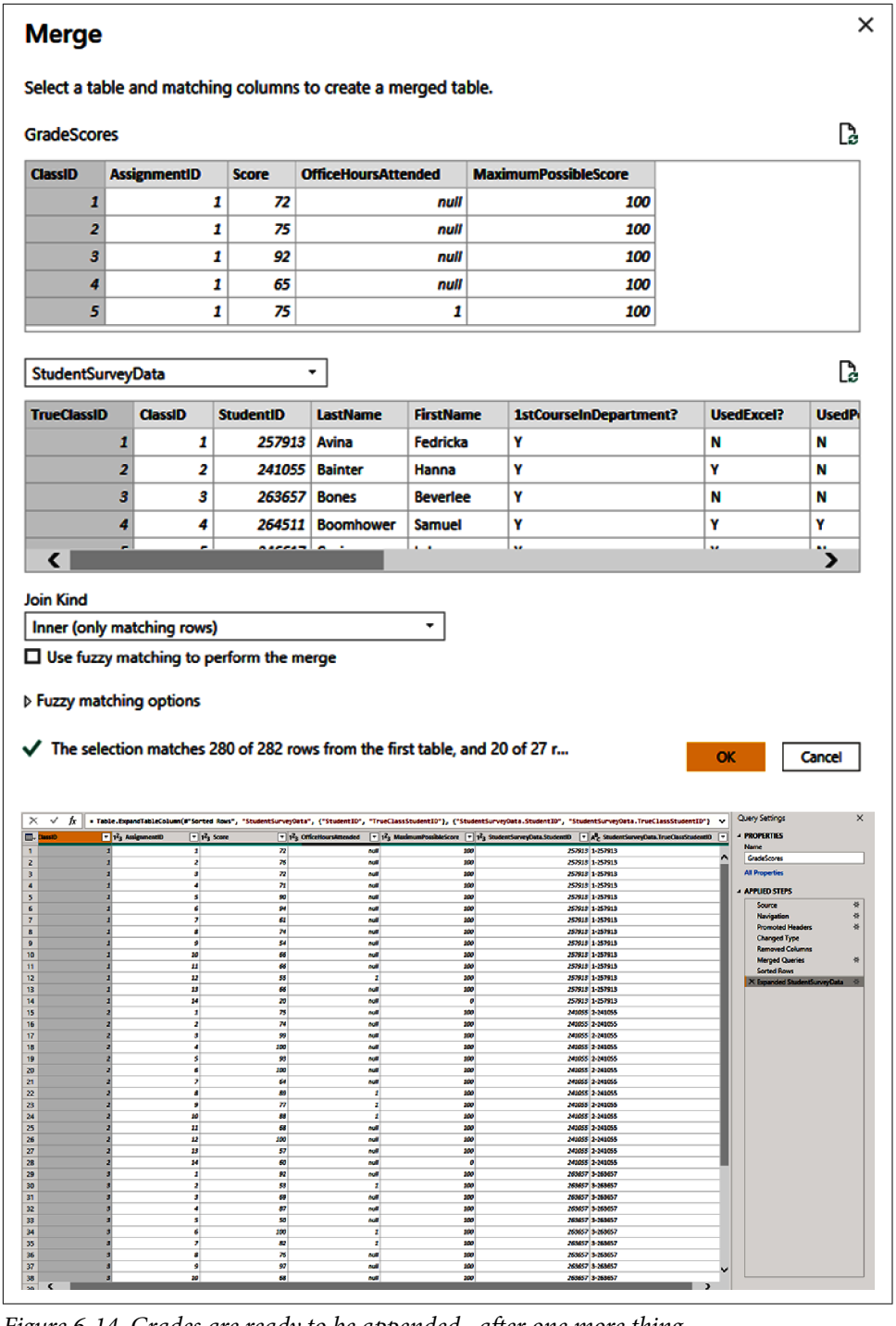


Figure 6-14. Grades are ready to be appended...after one more thing

Building Relationships

You might remember in Chapter 3 that my personal preference is to have Power BI *not* try to find default relationships. You can adjust this behavior from Power BI Desktop by choosing File → Options and Settings → Options → Current File → Data Load and unchecking the boxes for Relationships. If you do have that enabled, though, and you load your data, your Model view is going to look something like Figure 6-15. It's not super easy to read. It has a ton of inactive relationships, and we're going to hide more than half of these tables anyway.

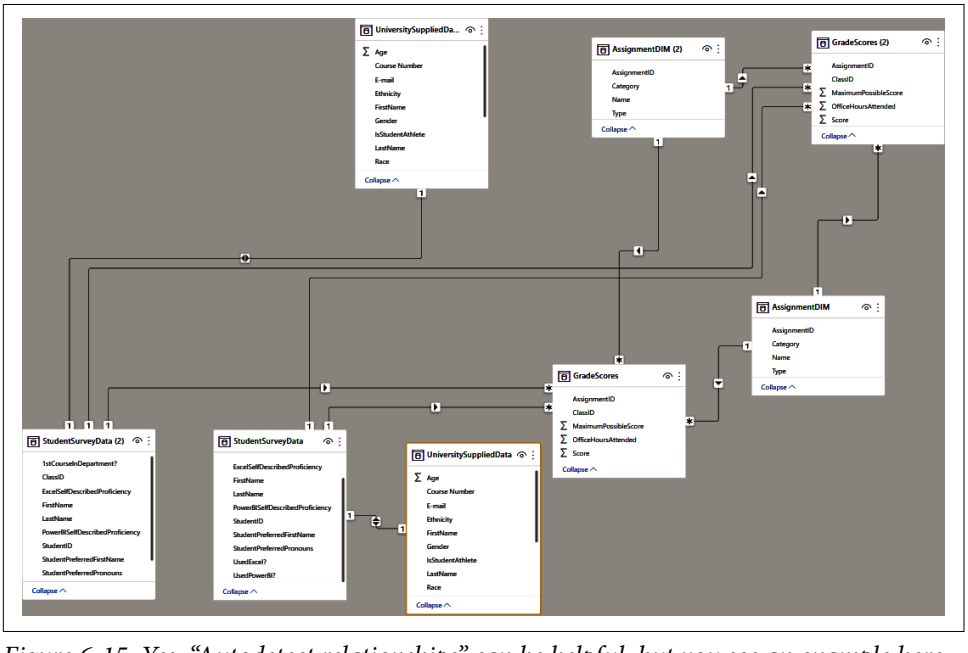


Figure 6-15. Yes, "Autodetect relationships" can be helpful, but you see an example here of when it isn't. I prefer not to enable "Autodetect relationships."

If you did have" Autodetect relationships" enabled, let's go ahead and remove all of them so we're starting from a fresh canvas together. You can delete relationships by clicking the line that shows the relationship and pressing the Delete key. Or you can go into the Manage Relationships section through the ribbon and remove the relationships from there.

Hiding Tables

As we just mentioned, we went through a lot of trouble to consolidate our data into the GradeScores, StudentSurveyData, and UniversitySuppliedData tables. The (2) tables are important in that they're called by Power Query and must exist, but that doesn't mean we need them for our analysis. For tables like this, which are functionally staging tables that we use to get the data into a certain shape so that we can work with it more easily, I like to go to the Properties window for these staging tables and move them to hidden status, as shown in Figure 6-16.

Now, it's important to note that "hidden" doesn't mean hidden from every view. It means it's hidden from the Report view. Therefore, you can still have tables that have relationships and serve important functions but not be available to report on.

However, what I want to do for now is, in the Model view, move all the (2) tables away from the core data model and make sure they're all hidden. You can see that when a table is hidden, it has a crossed-out eye next to the table name in the Model view. I'm going to collapse them to prevent looking at items I don't really care about, and I'll do that by clicking the Collapse button at the bottom of the table.

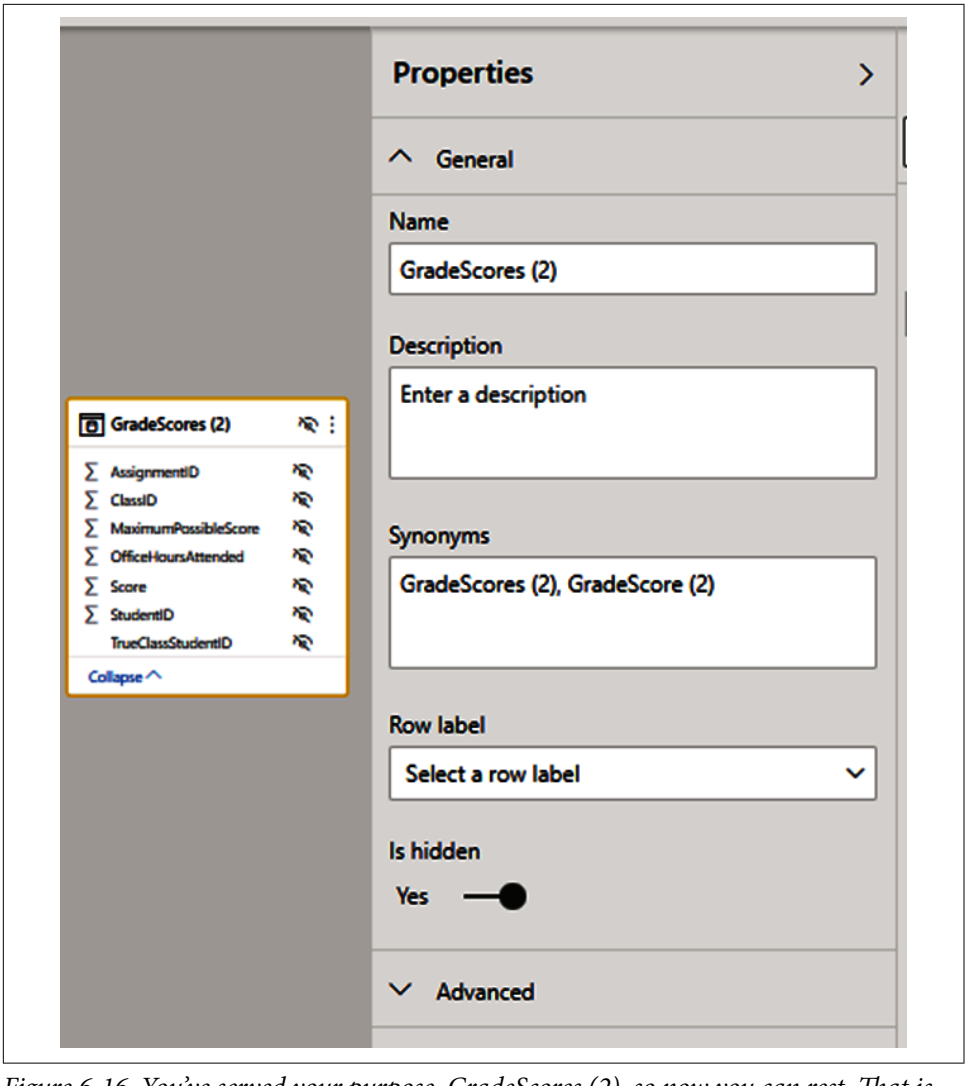


Figure 6-16. You've served your purpose, GradeScores (2), so now you can rest. That is done by hiding the data in the table.

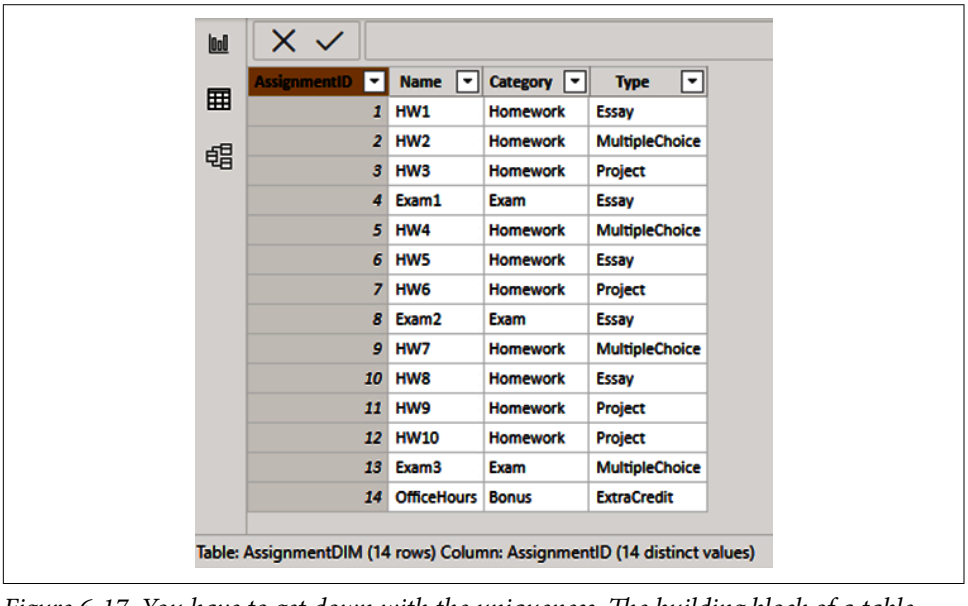
Identifying Our Relationship Columns

Now we can start working with our four core tables to build out a simple data model that we can use for reporting. We achieve that by creating relationships between tables.

Remember, these relationships serve as the pipes that allow data to flow from one table to another, connecting all the pieces together so that the internal query engine

can get all the data it needs to process a report request. Also remember that, as much as we can, we want a data model that uses as many single-directional relationships as possible while avoiding many-to-many relationships whenever possible.

To do that, we want to find columns that have only unique values, and we want to find that same column in our primary fact table—in this case, GradeScores. This would be our one-to-many relationship. In the Data view, as noted in Chapter 2, we can look at the table's data and observe different elements of the data, one of which is the ability to see how many distinct values and total values exist in a table for a given column. Let's take a look at AssignmentDIM this way in Figure 6-17.



AssignmentID	Name	Category	Type
1	HW1	Homework	Essay
2	HW2	Homework	MultipleChoice
3	HW3	Homework	Project
4	Exam1	Exam	Essay
5	HW4	Homework	MultipleChoice
6	HW5	Homework	Essay
7	HW6	Homework	Project
8	Exam2	Exam	Essay
9	HW7	Homework	MultipleChoice
10	HW8	Homework	Essay
11	HW9	Homework	Project
12	HW10	Homework	Project
13	Exam3	Exam	MultipleChoice
14	OfficeHours	Bonus	ExtraCredit

Figure 6-17. You have to get down with the uniqueness. The building block of a table relationship is having a column of unique values that relates to another column in another table.

As we see, when we select the AssignmentID column, in the bottom-left corner we see 14 rows and 14 distinct values. That's the column we are looking for. When we check GradeScores, we see that it also has an AssignmentID column, and those values line up with one another, with every value of AssignmentID in the GradeScores table being between 1 and 14.

Time to Get Building

We can create the relationship between AssignmentDIM and GradeScores in two ways. The first option is to use the Manage Relationships Wizard from the ribbon in the Model view. After we bring up the wizard and click New, we select our two tables

and identify the columns that will be the basis of the relationship by clicking the column names in each table (Figure 6-18). We click OK, and that relationship is built.

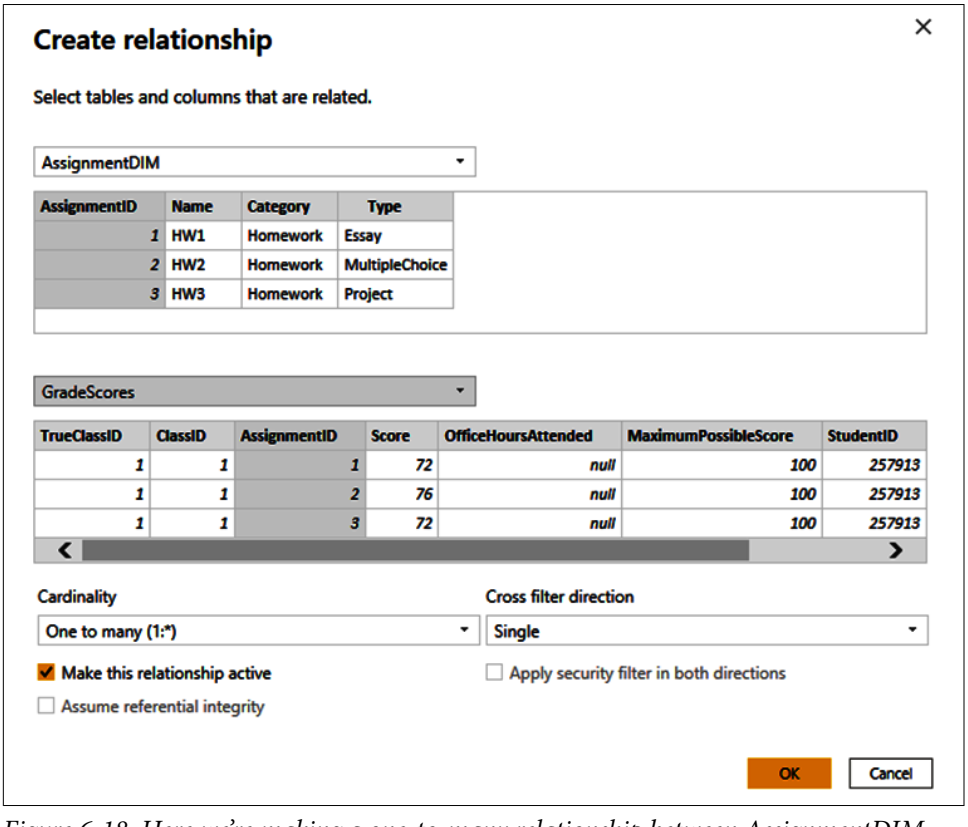


Figure 6-18. Here we're making a one-to-many relationship between AssignmentDIM and GradeScores on AssignmentID to AssignmentID

The second option to build relationships is, frankly, much faster. In the Model view, you click any column in a table, hold down your mouse button, and drag it to the column you want to build that relationship to in another table. Then you release your mouse button and, voila, relationship!

The downside to this approach is that it is easier to make mistakes by dragging from or to the wrong column. It's also more prone to error when you have tables with many columns and you must drag onto parts of a table that have to be scrolled through to find the right column; that can quickly get tedious, and that's when mistakes happen.

With AssignmentDIM connected to GradeScores, we need to connect StudentSurveyData to GradeScores, and UniversitySuppliedData to GradeScores as well. Thankfully, in our Power Query transformations, we got StudentID into GradeScores where

it wasn't there before. We can also see StudentID is unique in both of the two tables mentioned, so creating relationships on StudentID from UniversitySuppliedData and StudentSurveyData to GradeScores is simple enough.

After those relationships are built, we should have a core data model that looks like Figure 6-19. This gets us to a star schema of many dimension tables interacting with our fact table, surrounding it like a star. This has optimization benefits, but more importantly, makes the model incredibly easy to read and understand. You instantly know which tables can filter the fact table(s) you care about and how that filter behaves, either in a single direction or bidirectionally.

Figure 6-19. "When the line hits your eye like a star in the sky, that's amore." Our core data model is set up in a star schema, where many dimension tables surround a fact table.

Let's Get Reporting!

We've got our data in, we've got it transformed, and we've got it modeled. Now we can start getting those insights we're so interested in. There are tons of questions we could ask. From here, I'm going to focus on creating a first-level report page and some measures that will support our analysis. After you've gone through this chapter, think of some questions you might have from the data and try to find some answers! Now let's start with building that report page.

We Need a Name...

Here's some advice: I often find in reports that a simple thing like a title that reminds users what they're looking at can pay big dividends. This title could be something we reuse on multiple pages, identifying it as the title of the report, or in theory we could use different titles for each page. We could also have an entire report page that would serve strictly as some sort of title introduction.

Each approach has advantages and disadvantages, but, in this example, I'm just going to provide a simple title for the report at the top of the page. I'm going to do this with the classic text box. Just as we talked about in Chapter 2, the "Text box" option is under the Insert section of the ribbon in the Report view. I know that a text box is not terribly exciting, but it's helpful. As shown in Figure 6-20, I've put the text box at the top of my reporting area, center-justified it, and put some nice italics on it because I'm personally a sucker for italics. I like the Segoe font, as well as the default Power BI font, and I want it to be nice and legible, so I made it quite large.

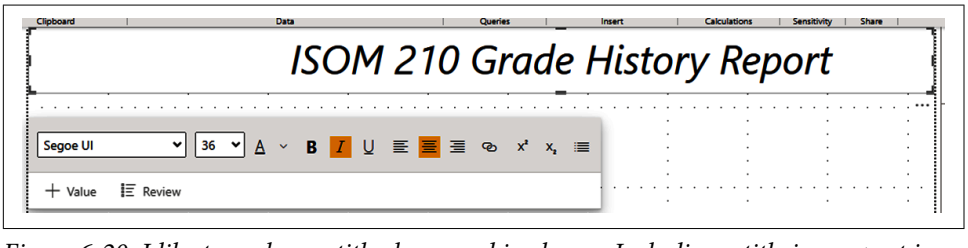


Figure 6-20. I like to make my titles large and in charge. Including a title in a report is a small touch that makes a big difference.

Cards Help Identify Important Data Points

Next, I want to highlight specific values that I think are valuable for understanding how our cohort of students performed and items that identify something unique about those students. I also like "lines" of information. I love blocks; I can't help it.

Our first card is a count of those students for whom ISOM 210 was their first class in the department. This uses the 1stCourseInDepartment? column in the StudentSurveyData table. However, if you make a card visual and drop that column into it, you'll probably see something like Figure 6-21.

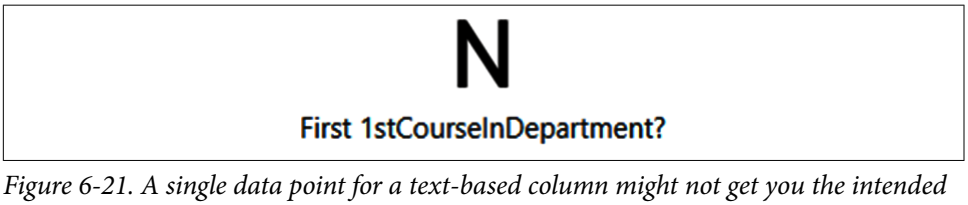


Figure 6-21. A single data point for a text-based column might not get you the intended result. N is not what we're looking for....

Let's note a couple things that are happening here. First, it's not a numeric column, so it does not have a default summarization. However, card visuals always display some summarization of data, and in this case, it is displaying the first result from the table that matches our filter context, which in this case is no filter context. So, we need to change the summarization from first to count. We do that by clicking the down arrow next to the column name in the Fields list in the Visualizations pane and then select the summarization we want—in this case, Count. Looking at Figure 6-22, we can see we're getting much closer now, but we're still not quite there.

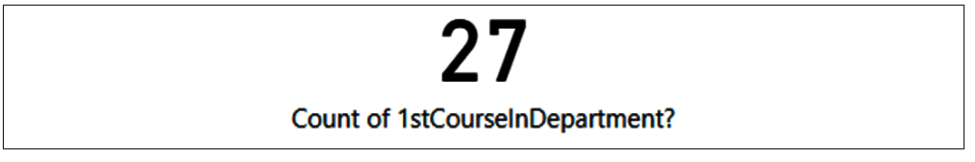


Figure 6-22. That's just a count of students! It's still not quite right.

So, we've got the card now displaying count instead of first, which is good. However, looking at the data in the table, we know there are 27 records in that table, so this is just a count of the nonblank records for this column.

We need to provide filter context. We want Power BI to give us a count, but we want a count of only the "Y" values. Currently in the Filter pane for this visual, we can see there's no real filter context.

Let's add some context by dragging the 1stCourseInDepartment column to the Filters on this visual area of the Filter pane and use Basic filtering to select only "Y" values. It is worth noting that the Count version of this field exists already since it's what we are displaying. We need to add the column to the Filter pane again to get its base values to filter as demonstrated next. When we do that, we get the 24 number we're looking for. Let's alias the column name by double-clicking the column name in the Fields section of the Visualization pane and renaming it to whatever you like. In this case, I chose 1st Course in Department Count. I now have the card I want and show the filter condition and aliasing, as you see in Figure 6-23.

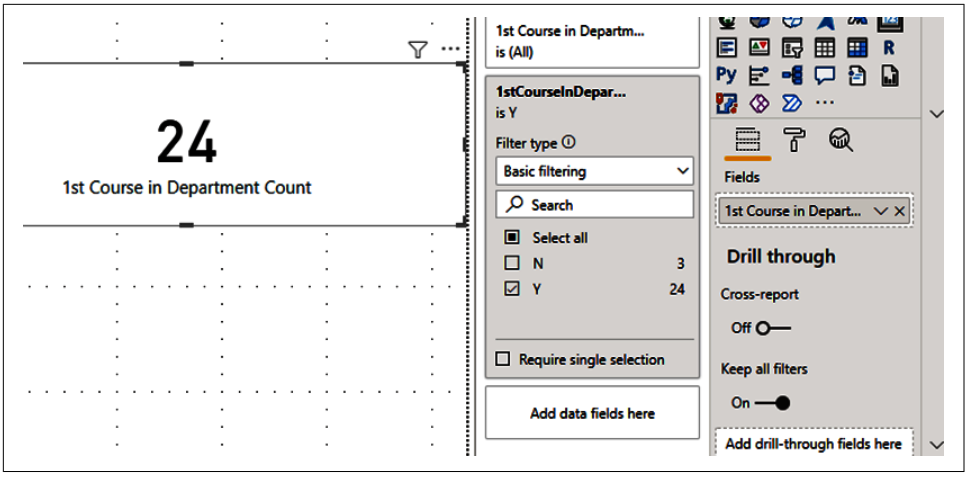


Figure 6-23. One card down, three to go!

The second card is very straightforward. Using the OfficeHoursAttended column in the GradeScores table, we can see that column is numeric and has a default summarization of sum. We make a card visual, drag that column into the Fields list, and that's about it. I did go ahead and alias the column name again so that would reflect in the card visual, so I had Office Hours Attended.

The third card is the Total Grade card. This should be able to calculate an individual's total grade for a given combination of assignments, as well as the entire class's total grade for a given combination of assignments. It's also important to note that assignment 14, which is extra credit, should be added to the numerator of someone's grade, but not the denominator.

We're going to need a measure here. The good news is that we did note a maximum possible score in the GradeScores table, and for assignment 14 that value is zero. Looking at the data in the table then, we should be able to sum up the Score and divide it by the maximum possible score to get the percent grade.

The one issue we have is that if someone were to choose only assignment 14, they would get a divide-by-zero error (`#DIV/0!`) because Assignment 14's maximum possible score is zero, and zero plus zero is zero. No problem. A DAX function that will be helpful in this case is called the *divide function*. Power BI does support the "/" operator in DAX to represent a divide function. However, if the result of that division would return an error, you're a bit out of luck. The divide function allows us to pass an alternate value at the end of the function that functions like an `IFERROR` clause. We can see the example DAX in Figure 6-24.

```
Total Grade =
DIVIDE (
    CALCULATE ( SUM ( 'GradeScores'[Score] ) ),
    CALCULATE ( SUM ( GradeScores[MaximumPossibleScore] ) ),
    "Extra Credit"
)
```

Figure 6-24. The `DIVIDE` DAX function allows us to automatically define a replacement value for a division error

Breaking down this DAX function, we start with a `DIVIDE` function that has a syntax of (Numerator, Denominator, Alternate Value). Our numerator is a `CALCULATE(SUM)` of `Score`. The denominator is a `CALCULATE(SUM)` of the `MaximumPossibleScore`. Finally, the alternate value is `Extra Credit` because I know the only time this would come back with a divide-by-zero error would be on Assignment 14, which is the extra-credit assignment based on office hours attended.

This gets us the result, but if we try to use it now, we're going to get a decimal result to two decimal places. That isn't necessarily bad, but typically we show grades as a percentage (at least we do in the United States; if it's different where you're from, do what would make the most sense to you).

I want this to be a percentage with zero decimal places, so I'm going to select the measure from the Fields pane that will show the Measure tools portion of the ribbon in the Report or Data view. I want the formatting section for this measure to look like Figure 6-25.

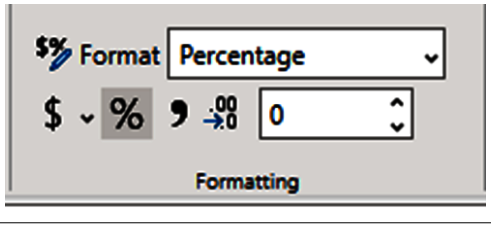


Figure 6-25. A little sprinkle of formatting dust

Now when we make our card visual and bring the Total Grade measure onto the Field list for that visual, we should see a card that looks just like that third card in Figure 6-27.

Our final card is the letter grade of that Total Grade. If your grade list was different from what I'm about to use, feel free to modify the DAX we're about to discuss to fit your needs.

To accomplish this, we need a DAX statement that will take the result of our Total Grade measure and provide a conditional result based on the result of that first value. In SQL, we could use something like a case when statement. In Excel, we could do a nested `IF` statement, and we could do that in Power BI too, but nested `IF` statements can get difficult to read very quickly, and in Excel and Power BI, nested `IF` statements aren't very performant.

To get the Total Letter Grade, I'm going to use a combination of the `SWITCH` function and the `TRUE` function. Often in Power BI or DAX, you'll see these two statements together and often referenced together as a `SWITCH/TRUE` statement.

`SWITCH` is a function that returns a value based on a condition. The example used in Microsoft's own documentation is a statement that takes the number of a month and offers the month's name as the switch value. So, 1 would be January, and 2 would be February, and so on. `SWITCH` calls first an expression, a value result of that expression, and then what to display instead if that value exists:

```
SWITCH(<expression>, <value>, <result>[, <value>, <result>]...[, <else>])
```

`TRUE` is a simple function that returns the logical value `TRUE`. However, a conditional statement can also be reviewed to see if the result of that value is true.

So, a `SWITCH/TRUE` function is basically saying that, for a list of conditions, when you get to the condition that is `TRUE`, switch to that value. In this case, my grade range needs to be between a given maximum and minimum value for a particular letter grade, so I will use the && operator to function as an `AND` operator. This can be useful alongside its `OR` operator equivalent of ||. The benefit these operators have over their explicit DAX `AND/OR` statement counterparts is that they're not limited to two passed functions, and I find them easier to read in statements like the one in Figure 6-26.

```
Total Letter Grade =
SWITCH (
    TRUE (),
    [Total Grade] >= 1.00, "A+",
    [Total Grade] < 1.00
        && [Total Grade] >= .925, "A",
    [Total Grade] < .925
        && [Total Grade] >= .895, "A-",
    [Total Grade] < .895
        && [Total Grade] >= .875, "B+",
    [Total Grade] < .875
        && [Total Grade] >= .825, "B",
    [Total Grade] < .825
        && [Total Grade] >= .795, "B-",
    [Total Grade] < .795
        && [Total Grade] >= .775, "C+",
    [Total Grade] < .775
        && [Total Grade] >= .725, "C",
    [Total Grade] < .725
        && [Total Grade] >= .695, "C-",
    [Total Grade] < .695
        && [Total Grade] >= .675, "D+",
    [Total Grade] < .675
        && [Total Grade] >= .625, "D",
    [Total Grade] < .625
        && [Total Grade] >= .600, "D-",
    "F"
)
```

Figure 6-26. Really not that bad! Here we're just providing what to display for a given result. Check result, display result.

We read the preceding statement as saying the following:

- Find the result of Total Grade.
- If that Total Grade is greater than or equal to 1, it's an A+.
- If that Total Grade is less than 1 but greater than 0.925, it's an A.
- Etc. etc.
- Sorry, you failed.

For each time Total Grade is calculated and this measure is on the canvas, it will figure out for that result which condition in the `SWITCH` statement is `TRUE` and then switch the `TRUE` statement for the Grade Score we've acquired.

Notice that for the calculation, I still used the decimal values for the conditional statements and not percentages. Even though we display the result of Total Grade in a percentage form, it's still calculated as a decimal value.

With that measure put together, we can simply drag the measure to the Field list of that last card visual, and our cards are complete! We can see our four cards together in Figure 6-27.

Figure 6-27. Our bottom row of cards gives readers quick, high-level information they might be looking for

Bars, Columns, and Lines

Let's put some pictures on this darn canvas already, right? I've got three visuals here, two bar charts and one line and column chart, and we will walk through those as shown in Figure 6-28.

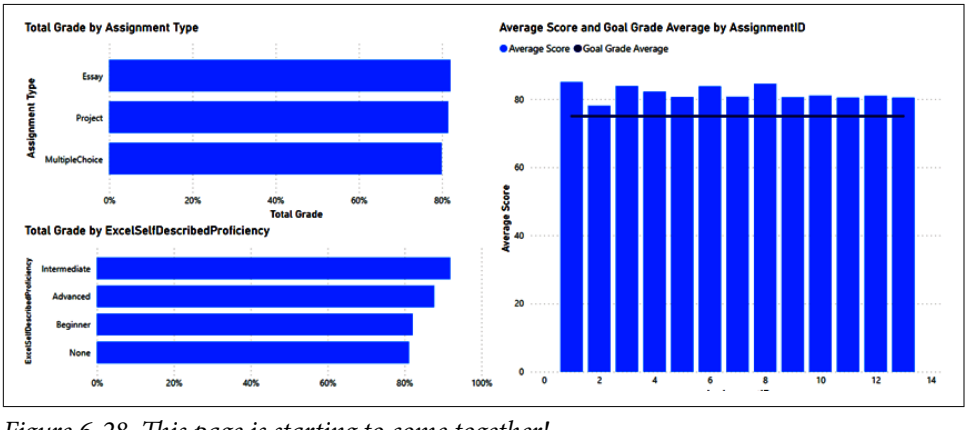


Figure 6-28. This page is starting to come together!

The first graph I'll deal with is the bar graph in the upper-left corner, Total Grade by Assignment Type. To add the visual to the canvas, I simply selected the clustered bar chart from the Visualizations pane and moved it over to the upper-left corner by clicking and dragging the outline of the visual to where I wanted it to be. The goal of this visual is simple: for each assignment type, I want to know the total grade. I put Assignment Type from AssignmentDIM into the Axis list (Y-axis area in more recent versions) and Total Grades into the Values list (X-axis area in more recent versions) in the Visualizations pane by dragging those columns from the Fields pane into the

relevant sections on the Visualizations pane with the visual selected. That was it, right?

The eagle-eyed among you who are trying this will probably see something quite strange! At first, this graph will show ExtraCredit, and it will be a blank bar! That's not very helpful at all. In fact, we know that Total Grade for ExtraCredit is going to be blank because ExtraCredit Assignments have a maximum score of zero, and as we discussed in the creation of our Total Grades measure, in a divide-by-zero situation, the measure should return "Extra Credit," which isn't a value we can display on a bar chart. So in the Filter pane, with that visual selected, you'll see that I've filtered out ExtraCredit, and that's how I get the result in Figure 6-28 for that bar chart, as shown in detail in Figure 6-29.

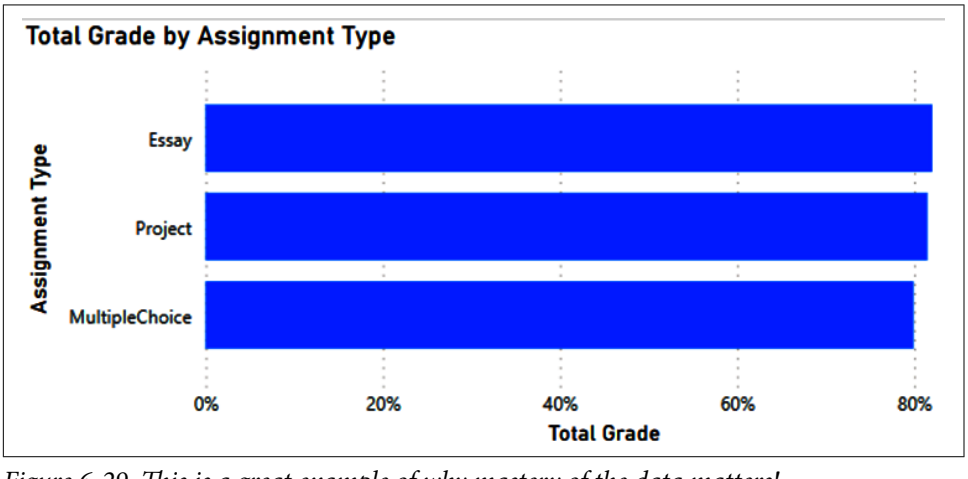


Figure 6-29. This is a great example of why mastery of the data matters!

That's straightforward. Let's go down to the second bar chart, Total Grade by ExcelSelfDescribedProficiency. This looks exactly like the first chart with a different axis, in this case the ExcelSelfDescribedProficiency column from the StudentSurveyData table.

I have done something different with this visual, though. I've added a second axis to make it drillable. I have two self-described proficiency columns in StudentSurveyData, one for Excel and one for Power BI. I put the PowerBISelfDescribedProficiency column under the Excel column in the Axis list. When I do that, I get the result shown in Figure 6-30. I could now use the drill options to drill into a specific combination of data between these two categories or show the results for every combination of these two columns that exist in the dataset.

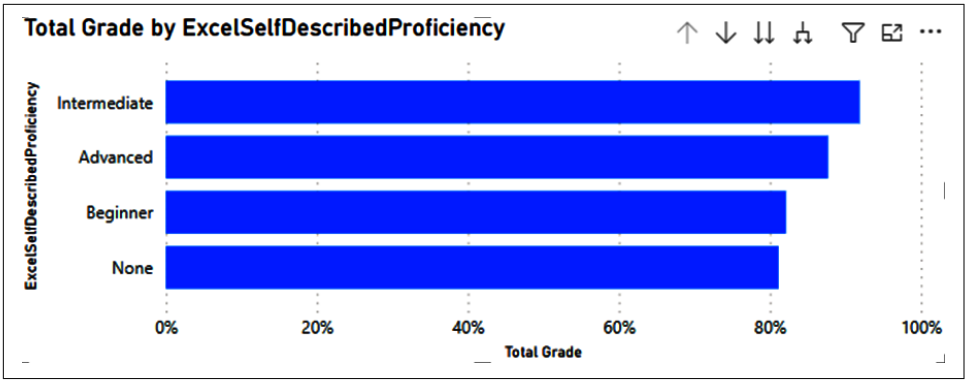


Figure 6-30. The ability to drill into a visual allows you to quickly change the dimension you're reviewing or combine dimensions for more categories to analyze

Finally, we have the line and column chart on the right side showing the Average Score and the Goal Grade Average with Assignment ID on the x-axis. In this example, I made a measure called Grade Goal Average, and it's literally equal to 75:

```
Goal Grade Average = 75
```

That's it. Nothing fancy, the measure equals a number that gets me this very convenient line.

Now, another way I could accomplish this same task would be to use a simple clustered column chart and use the Analytics section of the Visualizations pane to add a constant line of 75. I prefer the flexibility of the line and column chart, but it's important to note that there are multiple ways to accomplish something like this.

I want to fix one thing, though. We can see that Power BI is making a default axis, highlighting values at every even numbered interval. So, it shows 2, 4, 6, 8, etc., including Assignment 14, and it is showing this even though I currently have Assignment 14 filtered out in the Filter pane for this visual. I don't want the actual score to show, and I don't want it to show blank either. So, for that, I'm going to manually set the maximum axis value for the x-axis to 13. I do that by selecting the Range option under Visual for the X-axis and setting the Maximum to 13. I'll do that in the Format area of the Visualizations pane, and when I do that and have the visual selected, I can now see the result I want in Figure 6-31.

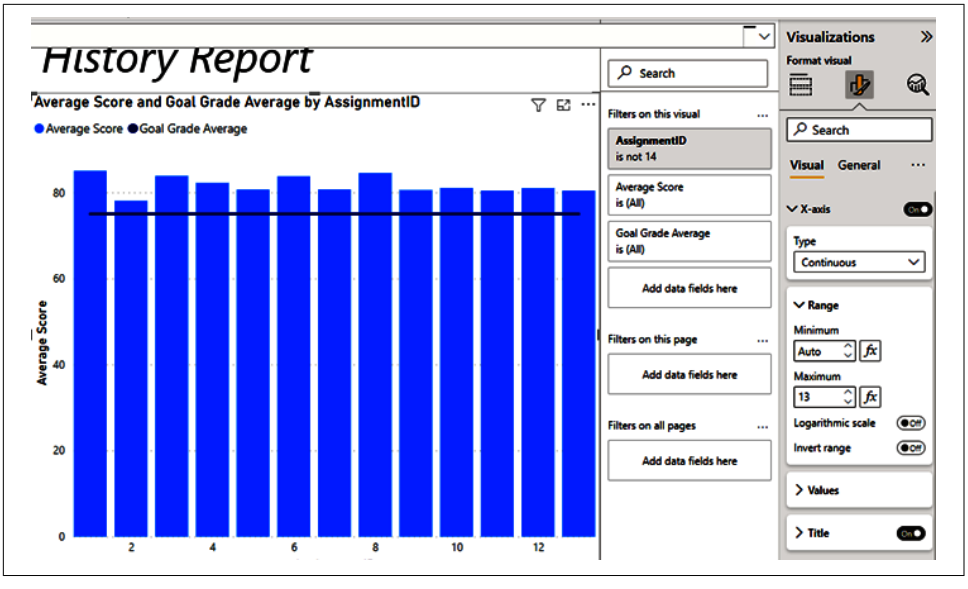


Figure 6-31. Better, much better

For interactivity, I add two slicers to the bottom-right corner of the report page so I can filter by Assignment Category and Term in the same way I add any other visual to the canvas. When I do that, I get a final first report page that looks like Figure 6-32.

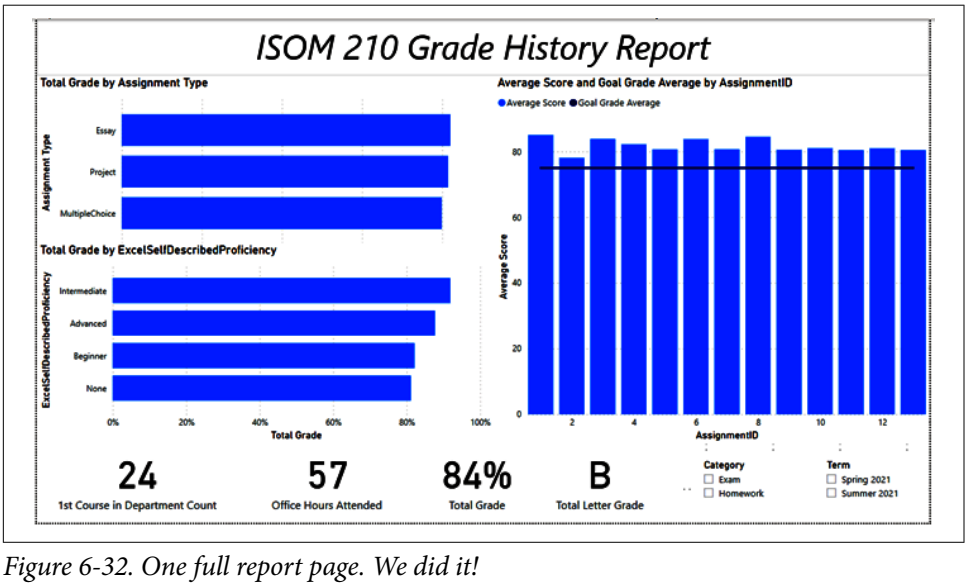


Figure 6-32. One full report page. We did it!

I'll leave you and this chapter with one conundrum that I'll let you figure out on your own. Using all the tools we've talked about, when you filter on Term, you'll notice that the 1st Course in Department Count won't filter. Identify why that's happening and figure out a way to solve the problem. You could take quite a few approaches, so get out there and experiment!

Conclusion

In this chapter, I wanted to go step by step to really demonstrate how we use the things we had talked about in the preceding four chapters. Now you should have a basic understanding of how to use Power BI Desktop to get data into the report, how to go about shaping it to your needs, building relationships and measures, and finally putting some results on canvas for users.

In the next chapter, we are going to transition a little bit to some more advanced reporting features in Power BI, which will round out the Power BI Desktop portion of this text. If you've gotten this far, I want you to know, you're doing great. You're pushing through on something new, and you should be proud of yourself for getting to this point. Let's keep pushing!

CHAPTER 7

Advanced Reporting Topics in Power BI

To this point, we have successfully loaded, transformed, and modeled data. We've built out an example report page using many individual pieces of Power BI Desktop that we discussed in the preceding chapters. In this chapter, we're going to review some of those pesky advanced topics that I said we'd address later.

The tools and topics we'll discuss here can be valuable and bring that little extra to your report that takes it from good to great or unlocks that one insight that makes the whole report click. However, using many of these functions without a firm understanding of the basics can also lead to misunderstanding exactly what you're doing or what you're looking at.

Specifically, we'll cover the currently available AI-powered visuals along with what-if analysis. I'll also discuss integration of R and Python into your Power BI setup and go over some of its limitations. These are incredibly powerful and have the ability to help you uncover facts about your data that would be otherwise buried and difficult to find. They'll also enable you to put forth scenarios to help your organization make better decisions.

AI-Powered Visuals

At the time of writing, Microsoft has put four AI-powered visuals into Power BI Desktop, and it's expected it will continue to invest in creating more AI-powered visuals to further extend Power BI's capabilities. All these visuals work in both Desktop and the service. You can find them in the Visualizations pane, shown in Figure 7-1. From left to right, the visuals are Key influencers, Decomposition tree, Q&A, and Smart narratives. Let's look at each one now.

Figure 7-1. AI-powered visuals as seen in the Visualizations pane. I promise these AI visuals have no connection to Skynet.

Key Influencers

The "Key influencers" visual is a powerful tool. It takes a value and analyzes it by a set of categories to determine varying levels of impact on a value. These categories are then identified and grouped to help you identify the relative impact compared to other categories.

You'll notice that if you try to use this visual on the dataset included in this book, it won't be very effective. I used several data randomization techniques to ensure that the data was as random as possible and used the "Key influencers" visual to test whether my data was sufficiently random. If it found correlations that could provide meaningful results, I went through another set of data randomization.

Since the data from our previous dataset is so heavily randomized, I will be demonstrating the "Key influencers" visual from a separate dataset from Microsoft known as AdventureWorks. AdventureWorks is a sample database shipped with Microsoft SQL Server. There are also many examples of AdventureWorks being connected to Power BI. In this example, I'm analyzing the count of invoices by territory name, product category, product subcategory, product description, and part number.

This visual can be split into several parts. Let's look at Figure 7-2, and we'll break down the visual from there. We're going to focus first on what impacts the count of invoices for the company. In theory, more invoices mean more sales, so figuring out what increases or decreases that count could be useful information.

We can go through the image from top to bottom and get an idea of what's going on. First, we can select either "Key influencers" or "Top segments." Currently and by default, "Key influencers" is selected.

Next, you'll see a framed question around what causes whatever we are analyzing to either increase or decrease. By default, this is set to Increase, but I selected Decrease here to show more results. Then you'll see on the left a list of selected conditions and how that specific condition impacts the value I'm analyzing.

In this case, when the subcategory is Touring Frames, the count of invoices for that subcategory is 43 less than the total average of all subcategories. This subcategory is highlighted, and you can see it farthest to the left in the column-graph portion of Figure 7-2. On the right, since the dimension in question is Subcategory, it shows me a graph that details the average number of invoices by subcategory and highlights the

specific group selected. You'll notice the graph on the right is also scrollable, allowing me to see all the subcategories in this way.

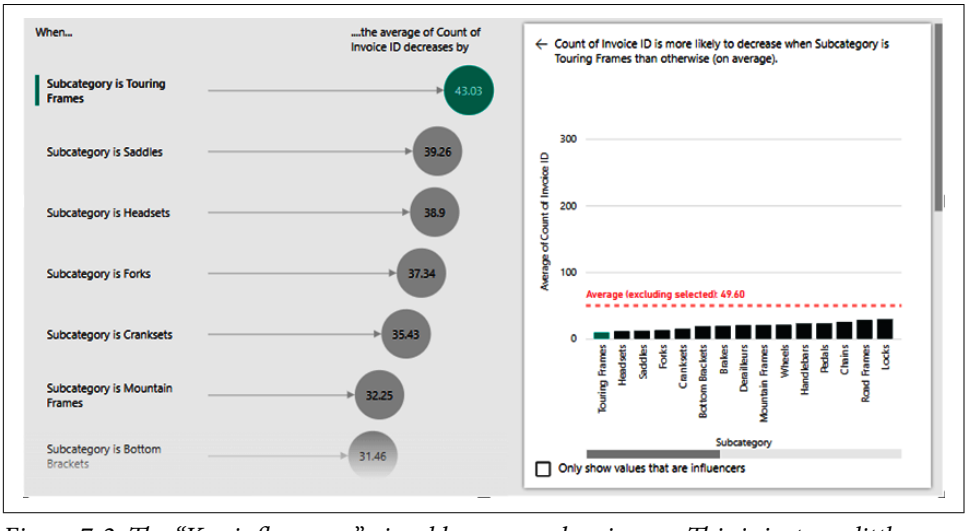


Figure 7-2. The "Key influencers" visual has so much going on. This is just one little piece.

Now let's go back to the top of the visual and select "Top segments." We can see what that looks like to start in Figure 7-3.

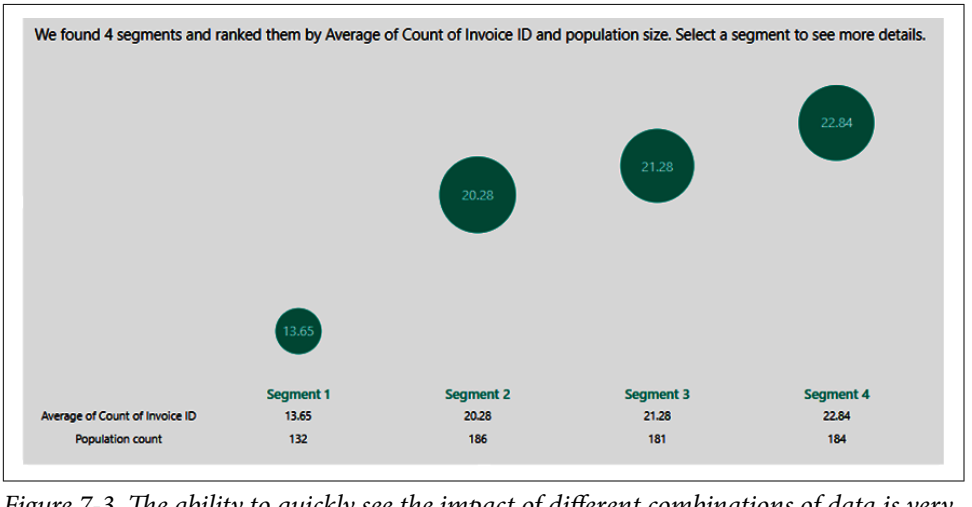


Figure 7-3. The ability to quickly see the impact of different combinations of data is very powerful

The first line is the same as before. The second line, though, is a little bit different. You'll notice that the question that frames the selection box is worded slightly differently. Instead of asking what might be causing a value to increase or decrease, it's asking when the point of analysis is more likely to be high or low. In this case, I have selected Low, but High is the default.

As the figure shows, Power BI has found four segments it thinks are interesting and worth diving into for this question; it displays a small scatterplot and details how it is organized. In this case, it's going from smallest average invoice count to highest, while also showing the count of records that make up that segment.

When you click into any of those bubbles (in this case, I'm choosing segment 2), you get to see all the details that make up that segment on the left. On the right side in Figure 7-4, you can see segment 2 in comparison to the overall value and see how much of the data is contained in the selected segment.

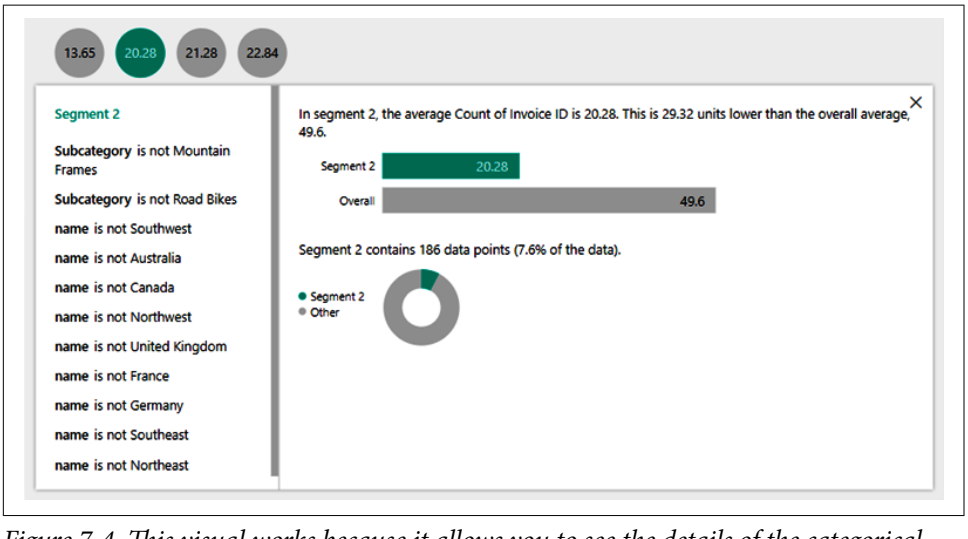


Figure 7-4. This visual works because it allows you to see the details of the categorical combination

While the visual itself is incredibly powerful, it can also be frustrating to use. As you add categories to the "Explain by" section of the Visualizations pane, the visual will refresh itself after every addition. It may struggle to find combinations of data that lead to the discovery of influencers or segments of note. It is not always more useful to keep adding categories either; the more categories you add, the smaller your population for each combination of categories gets.

Using this visual effectively requires you to really understand your data and to add some human intelligence. What does your intuition tell you should be important? Start there and see whether your intuition is validated. You'll find that to be a com-

mon theme in the AI visuals section. Human intelligence makes these things work; the AI visuals can't do it alone.

Decomposition Tree

"Decomposition tree" is significantly easier to understand than "Key influencers." For a given value, it shows you for each categorical level how that value is impacted by the values in that category. Let's return to our Cool School University dataset and put our Total Grade measure into the Analyze section of the Visualizations pane for this visual. What we see at first is pretty boring, as shown in Figure 7-5.

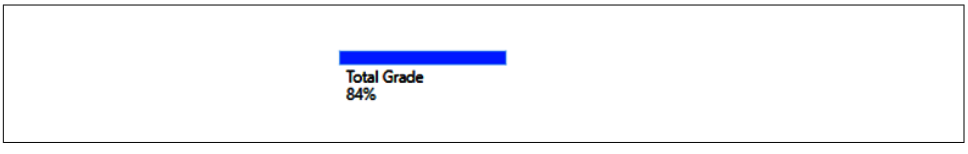


Figure 7-5. Is that really all there is?

This visual really shines when we add categories to the "Explain by" section. We can quickly see how that value changes based on the combination of values, and since it's a tree, we can have many "branches" to look at.

Let's first add Term from the UniversitySuppliedData table to the visual and see what happens...OK, that was underwhelming. You'll notice that the only thing different from Figure 7-5 is that there is a plus sign now next to the bar for Total Grade. When we click that plus sign, things start to happen. A small dialog box will appear with High Value, Low Value, or the name of the category. Selecting either High or Low Value will be accompanied by a lightbulb. In this case, I choose High Value, and we can see what that looks like in Figure 7-6.

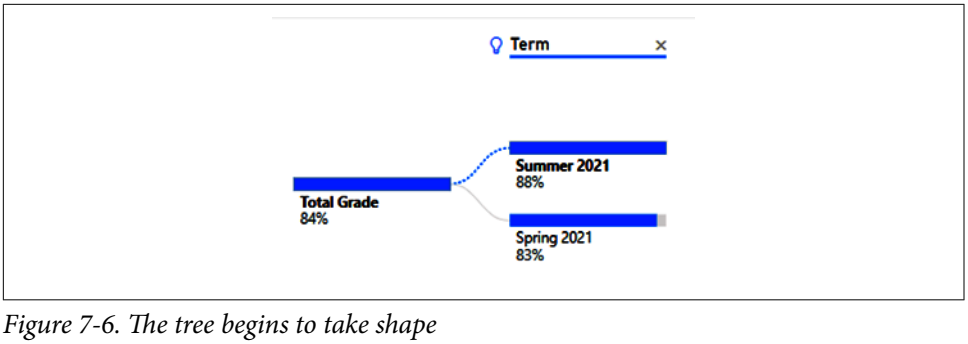


Figure 7-6. The tree begins to take shape

In our working example, we know from our work in Chapter 6 that our Total Grade is 84%, and we can quickly see that for Spring 2021 it was 83% and for Summer 2021 it was 88%. In Figure 7-6, we see that Total Grade has been broken into branches by term, Summer 2021 at 88% and Spring 2021 at 83%. The highest value for the

category is always shown at the top and is organized from high to low, so that's why Summer displays above Spring. If you hover over the lightbulb next to Term, you'll get a tooltip showing that Summer 2021 is the highest value.

If you want to remove a particular branch from the tree, you can click the X button to the right of the column name on the visual. This removes the branch from the tree, but not from the visual. This is important because you can rearrange the branches of a tree at any time by removing a branch or branches and then choosing a different column order when the plus sign at the end of the tree is selected. Take note that any value in the tree is going to be reflected at whatever level the selected value of the category is and all the selected categories before it. Let's see what I mean here by looking at Figure 7-7.

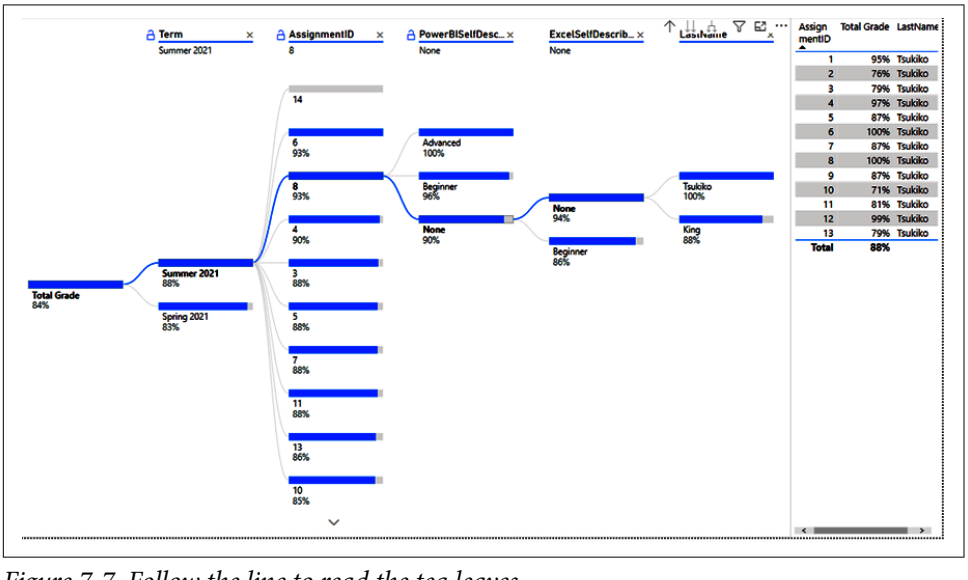


Figure 7-7. Follow the line to read the tea leaves

Take a look at the ExcelSelfDescribedProficiency category, which is second from the right. For the selected value of None, it has to go through the flow shown by the selected values, and this is demonstrated by the line that goes through each category.

So when we say that people who described their Excel proficiency as None have a 94% total grade, that is true only for people who also had no Power BI proficiency specifically on Assignment 8 who were in the Summer 2021 course. In other words, to read a data point in this visual, it's still dependent on all the parts of the visual that precede it. Each branch, when selected, shows only values that are also true for all the categories preceding it.

So, when we look at the LastName category, only two students had no Excel or Power BI expertise in the summer 2021 class for Assignment 8. Ms. Tsukiko did a very good

job on the assignment and got a 100%! Mr. King didn't do so badly at 88%. At a glance, we can see that, on this assignment at least, people who had no Power BI or Excel proficiency in aggregate did better than those who had no Power BI proficiency but did have beginner Excel proficiency.

If you ever get lost reading this visual as you go through various steps, remember to follow the blue line through the categories and remember each step is dependent on the step before it.

Two other things to note for this visual. First, like all Power BI visuals, it does interact with other visuals, but only when a value on the last branch of a tree is selected. Otherwise, the visual will filter itself to show items selectable for the next category. Second, this visual has no scroll bar, but you can click and drag right to left to scroll horizontally. There is no vertical scrolling, though, so to navigate vertically you have to use the arrows like those shown at the bottom of Figure 7-7 in the AssignmentID category. Note to Microsoft: please give us easier vertical scrolling capabilities.

Q&A

The goal of the Q&A visual is to empower end users to ask questions of the data by using natural language. Power BI will attempt to take your question and turn it into a chart or set of results that answers that question.

Q&A can be powerful, but out of the box it can feel like it has a couple of screws loose. Some of the suggested questions might not seem relevant, and while Microsoft continues to improve on Q&A in seemingly every release, I wouldn't call it quite "natural language" yet.

However, you can do many things to provide just enough extra context to Power BI to help make Q&A feel more natural. Plopping down a Q&A visual into our canvas, we first see the result in Figure 7-8.

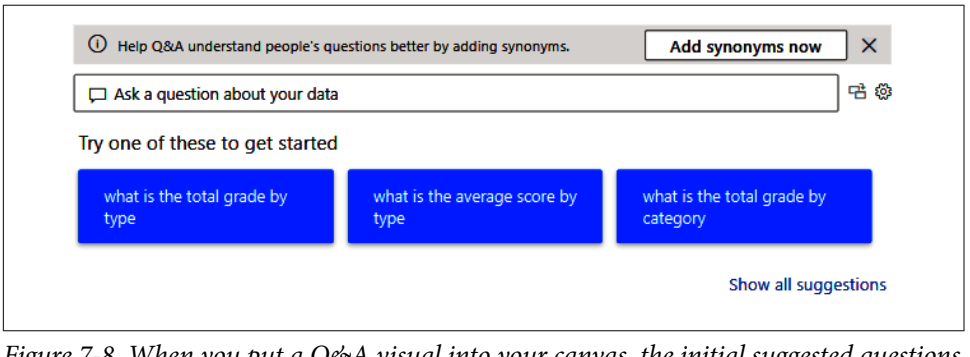


Figure 7-8. When you put a Q&A visual into your canvas, the initial suggested questions aren't always super exciting

Just looking at the suggested questions, we can see that Q&A does a good job of recognizing our field names, but those field names might not be obvious in what they mean to other users. For instance, total grade by category means, "What is the total grade by the category column?" You and I both know that column exists only in the AssignmentDIM table, but someone else might not. Further, what does "category" mean? We both know that's the descriptor for whether an assignment was homework, an exam, or extra credit. But does our audience necessarily know that?

From this example, we can see one of the issues with Q&A and the example questions it generates. At first, it has only our column names and relationships to work with. It will apply some simple aliasing where it can, but that won't necessarily get us all the way there.

However, we can still use one of the example questions to demonstrate what Q&A can provide as a result. In this case, I'll select the first question on the left, "What is the total grade by type?" In this case, Power BI generates a bar chart showing the total grade measure for each type of assignment. I can see that result in Figure 7-9. Notice that this visual interacts with all other visuals on the canvas and can be interacted with by other visuals, like anything else. So keep in mind what question you asked when you start cross-filtering across visuals with Q&A!

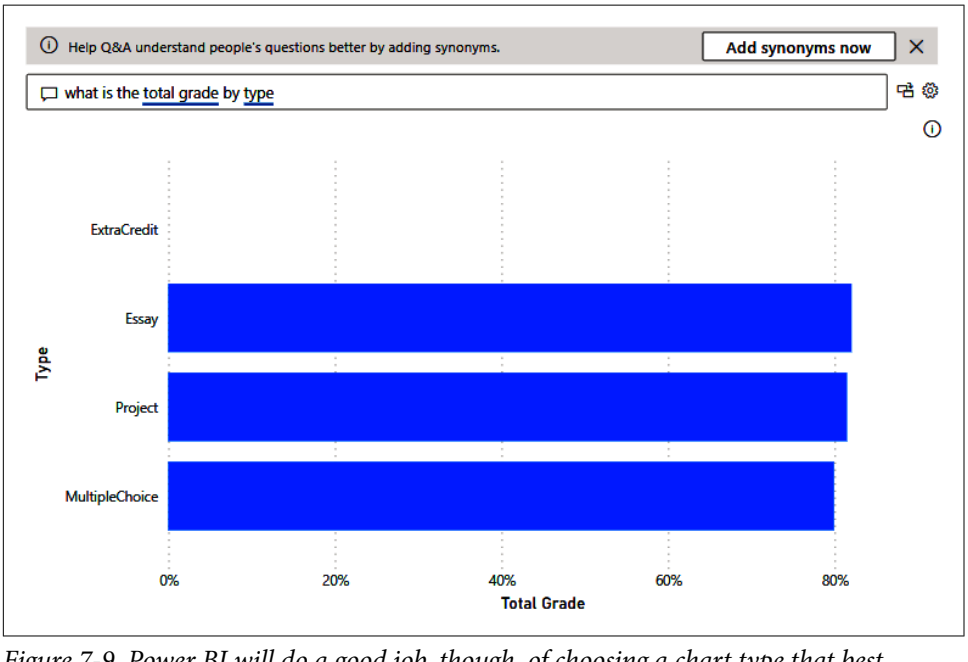


Figure 7-9. Power BI will do a good job, though, of choosing a chart type that best answers the question

Breaking down the visual itself, there will always be the option to add synonyms unless you click the X next to the button. We'll get to that in a little bit. I appreciate that it keeps the question you asked in the text bar, and it highlights the keywords it used to identify the intent of the question—in this case, "total grade" and "type." Let's say you find a result you really like from Q&A and want to make it a more permanent fixture. You can click the first button to the right of the text box to convert the Q&A visual into a visual of the type generated. The widget next to that leads you to Q&A settings. The circled "i" provides a quick tooltip when hovered over, confirming what the visual is showing.

Since the question remains in the text box, we can easily add, change, or remove a part of a question. For instance, I know why ExtraCredit shows as blank for Total Grade based on the way the measure works. I'd rather remove it. I'll add "without ExtraCredit" from the visual and voila, it's gone!

When you make additions to a question, the text box will show a drop-down that will attempt to guess your question via a search engine. This feature, as you might expect, gets better the more it is used and the better your synonyms are.

We could even add another step to this question and add something like "for last name King." Then we can see those results for students with the last name King. One of the things Q&A does well is allow you to refine your question. We started with total grade by type, removed or filtered out the ExtraCredit type, and then specifically filtered the result down again for those students who have the last name of King.

We could rename columns to more friendly names if we planned on leveraging Q&A extensively. That might help the AI more easily figure out what a column means. We can also add synonyms to certain column descriptors to help make the language easier when users type questions. In Figures 7-8 and 7-9, a button in the top-right corner says, "Add synonyms now." If we click that, it will open the Q&A setup window and automatically navigate us to the Field synonyms section, as seen in Figure 7-10. Note that it will show all the tables in the model, even the ones we have hidden in the Report view. However, by default it will remove those tables from Q&A selection.

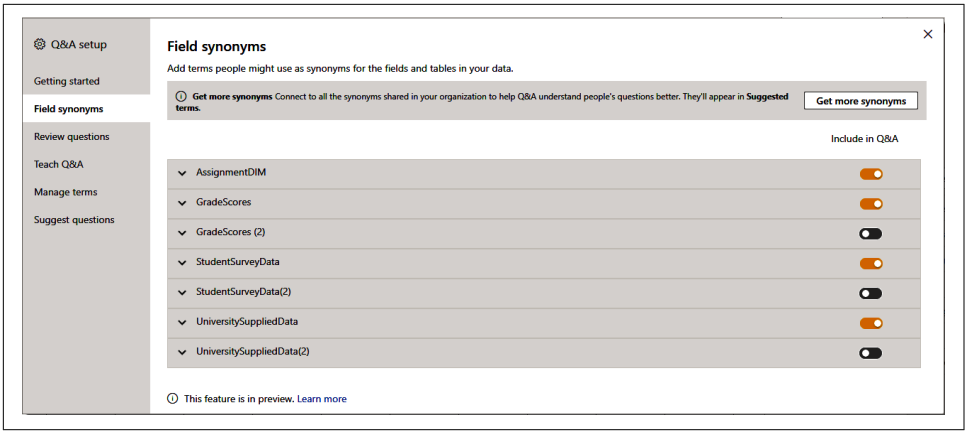


Figure 7-10. If you're going to make Q&A shine, you're going to need to spend a lot of time here in the Field synonyms section

When you choose a table, you'll see the list of columns under Name, the currently recognized synonyms under Terms, and a list of Suggested terms to the right of that. Finally, you will still see a toggle at the column level to let you include or not include that column in Q&A results. In Figure 7-11, I've selected the UniversitySuppliedData table to expand.

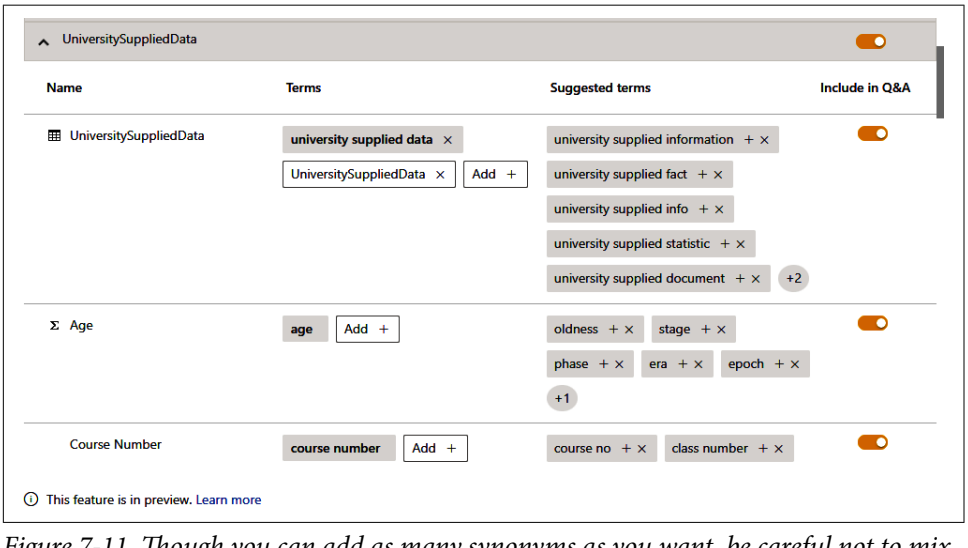


Figure 7-11. Though you can add as many synonyms as you want, be careful not to mix and match them too much

If I click the plus button on any of the "Suggested terms," that term will immediately be added to the list of terms for that column. Thinking about this process from right

to left best demonstrates how this works. "Suggested terms" provides a list of possible synonyms that become Terms. When one of those suggested terms is selected, it is added to "Terms." Those terms are then the actual synonyms that Power BI will use when it encounters a question in Q&A to figure out which column it should go use to answer the question. The object referenced by a given term is shown under the Name column. Terms can reference table names, column names, or measures. You'll also notice that Power BI will have some basic terms already preset, but they're not particularly insightful. For instance, the column FirstName will have a Term preset of first name.

I would love to be able to give you advice on setting up Q&A in your own enterprise; however, language is unique to an organization. Q&A is one of those features that Microsoft continues to improve on little by little, and is committed to natural language querying in Power BI.

My suggestion before deploying Q&A is to throw as many questions as you can think of against the engine. See what results come back, add synonyms where necessary, and try again. Lots of opportunities exist for fine-tuning results, and improving Q&A in your organization is going to be an iterative process.

I want to touch on two other features of the Q&A setup: the "Review questions" and "Suggest questions" features. The "Review questions" section allows you to review every Q&A question that has been submitted across all the datasets you have access to and that has been asked in the last 28 days. Annoyingly, it shows you all the datasets you have access to, not just the datasets you have access to that have had questions asked against them with Q&A. However, if someone does have a question, you can review it and the answer Power BI provided, to check for accuracy and provide guidance on how you would have preferred the answer be given for that particular question.

"Suggest questions" allows you to put questions in front of your audience, like the ones we saw when we first opened the visual. The difference is that those are questions you have already reviewed. Think of these suggested questions as easy-to-use cheat commands for other users building reports using Q&A visuals. You know what results will be displayed when that question is selected. These can be great starting points for your organization to begin using Q&A in a bit more of a sandbox, give confidence to users about the results, and encourage exploration.

This can also be good to provide a specific set of answers to specific questions. In a single visual, users can choose suggested questions that may help them frame other visuals on a given report page. Q&A is a complicated beast, and if you want to get the most out of it, it's just like getting to play at Carnegie Hall. It takes practice, practice, practice. If you would prefer not to allow users to use Q&A against a dataset in the Power BI service, you can also disable Q&A against datasets in service.

Smart Narrative

"Smart narrative" as a visual is a bit different in that it doesn't do anything on its own. If you try to put this visual onto a blank canvas, an error message will appear, and a text box will be generated instead. However, when added to a report page with other visuals on it, the "Smart narrative" will "read" the other visuals from the data points of those visuals and construct a narrative to help "read" the data.

In Figure 7-12, you can see I took our final page from the end of Chapter 6 and made some room for a "Smart narrative." It's important to note that a "Smart narrative" visual can be cross-filtered and interacted with by any other visual on the canvas and will update its narrative accordingly! Also, because it is still inside a functional text box, you can add commentary to the "Smart narrative" that will be present regardless of filter context changes that would influence the rest of the narrative.

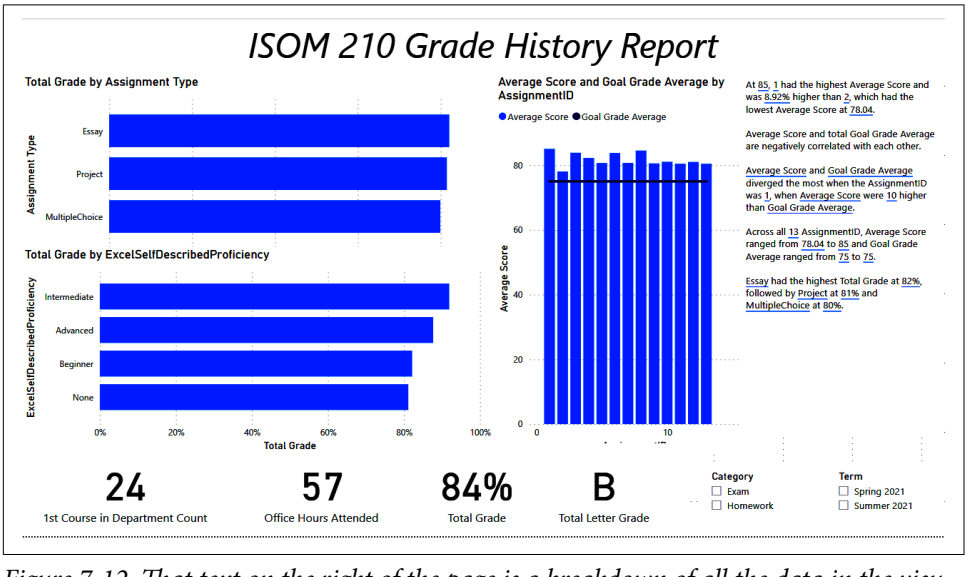


Figure 7-12. That text on the right of the page is a breakdown of all the data in the visuals on this page

The "Smart narrative" is a double-edged sword in my view. On one side, it's an absolute godsend for data literacy. This visual does a good job of breaking down the data in a way that can make it readable for nontechnical users and others who aren't as familiar with the data, and that can have so much value for report users. However, it is also quite big and clunky. At the end of the day, it's a large text box that can produce narratives so long that you need to drag them up and down to read the whole thing. Design space is at a premium, so whether or not the "Smart narrative" visual is the right choice is ultimately contextual to your audience and your design choices.

What-If Analysis

In many scenarios, you ask yourself a question that starts with two simple words: "what if." What if our production line gained 1% of efficiency? What if we had sold 200 more units? What if our students had scored 15 more points over the course of the semester? Each of these questions is the basis of *what-if analysis*. This is the practice of taking a value and modifying it by a parameterized result and seeing what changes.

What-if analysis in Power BI begins with the creation of a What-if parameter. This is done from the Modeling tab of the ribbon in the Report view, under the "What if" subsection. Click the "New parameter" button and you'll see a dialog box pop up, as shown in Figure 7-13.

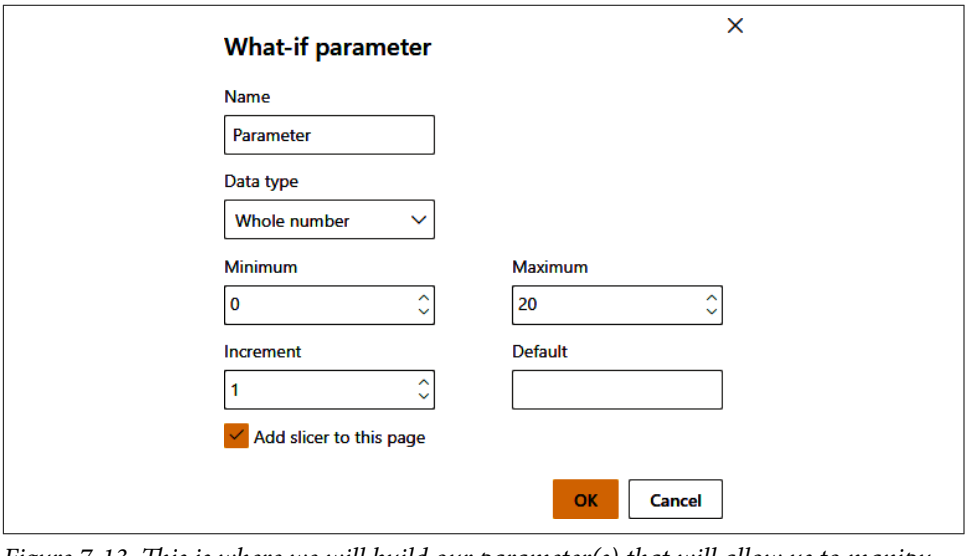


Figure 7-13. This is where we will build our parameter(s) that will allow us to manipulate values

Parameter Setup

The What-if parameter has several inputs that need to be managed. Name is obvious but also has a unique consequence. Whatever name is selected, a couple of objects are going to be created in the data model to support the parameter.

The finalized What-if parameter will use the `GENERATESERIES` statement with the name of the parameter to create a DAX-generated table. It will then create two objects inside that table. One will be an object that looks like the "New parameter" button that we clicked to bring up the dialog box. This will be a slicer that can be put onto the canvas to manipulate the value of the parameter on the report canvas.

The second thing that will be created is a measure that uses the `SELECTEDVALUE` function so that you can call in another measure for this parameterized value to use in other DAX functions. This will be demonstrated in a little bit.

After naming our parameter, we need to define the type of data it can be: Whole number, Decimal number, or Fixed decimal number. A *fixed decimal number* has a fixed location for the decimal separator. This can be useful in specific scenarios where rounding could introduce errors, or you have really small numbers that you are modifying, and errors could accumulate. In our case, we don't hate ourselves enough to grade scores to decimal points, so we can go ahead and choose "Whole number."

The minimum and maximum values set the first and last value in the `GENERATE SERIES` statement, which will create the table and values that will be utilized by the parameter. I personally like to keep the value of zero somewhere in the series. It doesn't have to be the minimum or maximum, but having zero in the series can be helpful when I want to show the baseline value without modification. In this case, I will choose –100 for the minimum and 100 for the maximum.

Next, I set the increment for the change. This describes the allowed interval for editing the value. For instance, if I chose 5, with a minimum of –100 and a maximum of 100, I could choose –100, –95, –90, ... 95, 100. I can't choose 97 or 43 or –37. If my increment was 1, I could choose any whole number value between –100 and 100. Obviously, if you are going to have a decimal-based increment, you should be using a decimal or fixed decimal data type. For the purpose of this example, I am going to set an increment of 1.

Finally, I set a default value. This is the value that will be shown when I first bring the slicer from the Fields list onto the canvas, and it will be the value that the parameter will reset to if someone uses the "Reset to default" feature in the Power BI service. When zero is selected as the default, the slicer will initially show a blank value. This is nothing to worry about and is intended behavior. We have one final selection to make, and that's whether we want Power BI to automatically add the slicer it will create to this report page.

DAX Integration of the Parameter

After clicking OK, I now have a parameter. However, that parameter doesn't do anything! That is because nothing can currently be modified by the parameter. We have to call the parameter in some value that we will bring onto the canvas. This is why the measure created along with our parameter is so important. With the slicer on the report page, we can modify one of our measures to add the value of that parameter to the total. Let's go back and review our Total Grade measure and then show where we could place the created measure from the What-if parameter in Figure 7-14.

```
1 Total Grade =
2 DIVIDE (
3     CALCULATE (   SUM ( 'GradeScores'[Score] ) ) ,
4     CALCULATE (   SUM ( GradeScores[MaximumPossibleScore] ) ),
5     "Extra Credit"
6 )
```

```
1 Total Grade =
2 DIVIDE (
3     CALCULATE (   SUM ( 'GradeScores'[Score] ) ) + [Total Grade Modifier Value],
4     CALCULATE (   SUM ( GradeScores[MaximumPossibleScore] ) ),
5     "Extra Credit"
6 )
```

Figure 7-14. The What-if parameter creates a measure that we can call, just like any other in a DAX statement

With this modification in mind, based on our dataset, I'm going to focus on the following question: What if our total grade was *X* amount higher or lower? What would that do to the relevant letter grade? For the purpose of this demonstration, I'm also going to revert the Total Letter Grade measure to display two decimal places instead of zero.

I've created a very simple page here with a term slicer, the What-if parameter slicer, and a table with the total grade and total grade letter. Let's move the grade modifier to –100 and to 100 and see what that does to our table in Figure 7-15.

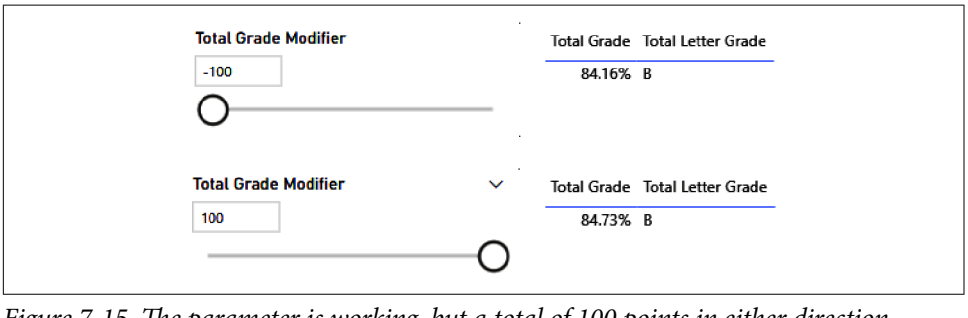


Figure 7-15. The parameter is working, but a total of 100 points in either direction doesn't seem to make a big difference

We can see from Figure 7-15 that those 100 points either way don't make or break the class. However, given that we are talking about grades, you could imagine that if we shrink the population, that difference would be more obvious. We know previously that Summer 2021 had only seven students, but their overall grade was higher. Would 100 points get Summer 2021 up a letter grade? Let's try expanding the analysis here to see what would happen in Figure 7-16.

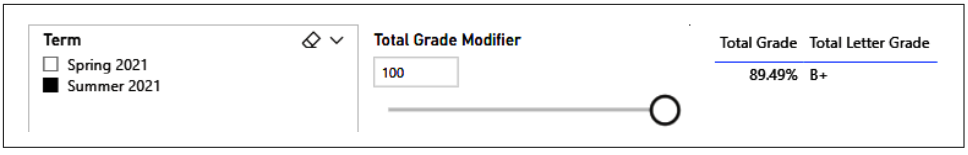


Figure 7-16. Fate is a cruel mistress. It seems they needed 101 points.

Well, if we added 100 total points of score to the summer 2021 class, they would have finished with a total grade of 89.49%, which is 0.01% away from rounding up to an A–.

Parameter Modification

However, what if we wanted to expand our parameter? What if they needed 1 more point or 10 more points? We know it's greater than 100. The good news is modifying the minimum, maximum, and increment of a What-if parameter is extremely simple. You just modify the DAX that created the table of values that are used.

In this case, we will modify the `GENERATESERIES` statement. This statement follows a very simple statement logic: `GENERATESERIES ( Minimum value, Maximum value, increment )`. You'll notice with a minimum of –100, maximum of 100, and increment of 1, our `GENERATESERIES` statement looks like Figure 7-17. We will modify the middle part of the statement to 101, just to add that one extra point to allow our parameter to go up to that value.

```
1  Total Grade Modifier =
2  GENERATESERIES ( -100, 100, 1 )
```

Figure 7-17. What-if parameters can't work without having an actual value in the data to grab, and that's what the series does for us

After that small modification to the maximum value, we can see in Figure 7-18 that, had the summer 2021 class gotten 101 more points of score, the class would have had an aggregate grade of an A–.

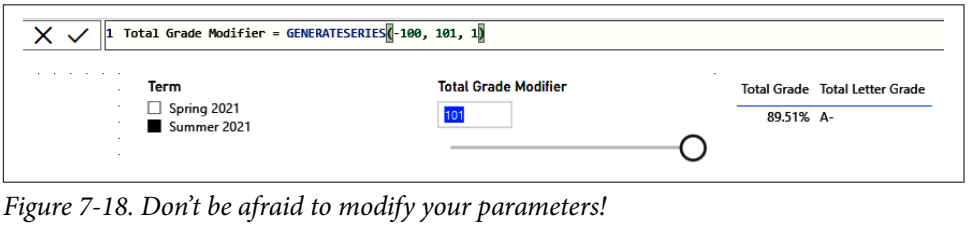


Figure 7-18. Don't be afraid to modify your parameters!

The ability to modify these parameters is incredibly helpful. You can also create multiple parameters and have them interact. For instance, let's say your widget factory

produces 100 widgets at $10 each. You know how much revenue that is. However, you could have What-if parameters for both price and quantity and try different combinations of price and quantity to see if one is more impactful than the other, given a set of operational constraints.

What-if parameters are incredibly powerful and flexible. I've seen some crazy accountants get all the way down to the General Ledger Code level with their What-if parameters.

In baseball, you could add or subtract *X* number of hits and figure out what the change to batting percentage would be, for an individual or a whole team.

Personally, I have a Power BI report that estimates my total grocery spending each month, and I use What-if parameters to estimate the impact to my wallet based on possible rates of inflation. It is one of the most powerful tools in the Power BI toolbox.

R and Python Integration

R and Python are both programming languages that can be used to extend the analytics capabilities of Power BI. R has its roots in statistical programming, and Python is a very flexible scripting language. Both languages have a wide variety of applications around statistics, data visualization, data wrangling, machine learning, and more.

While learning how to program in R or Python is beyond the scope of this text, learning how and where you can integrate these functions into your Power BI work and the implications that come alongside using R and Python in Power BI are pertinent to our discussion.

Limitations of Using R and Python

I know this feels like a weird place to start, but utilizing R and/or Python comes with some serious consequences that users should be aware of. First and foremost, if you have R or Python being used either in Power Query or for visualization, you will be able to refresh a dataset utilizing these tools only with a personal data gateway, which must be installed and actively running on the machine at the time of refresh.

The Power BI Data Gateway is a tool that can be installed to allow access to on-premises data from the Power BI service. It can also be used in "personal" mode, which allows the gateway to run on your local machine. For R and Python, refreshes must be done using a personal data gateway, as it's your machine that is known to have the necessary libraries or packages to run the request. The Power BI service does not have installations of those languages in a place that can be accessed universally by datasets. From the perspective of Power BI Desktop, this is just an annoyance. However, for the dataset in service after publishing, it can be a significant limitation.

Next, it's important to understand that Power BI does not support every R library or Python package. When you are doing development on your local machine, you have access to all the resources you have locally installed.

For instance, if you had every R library ever known to exist on your local machine, Power BI Desktop can use those locally just fine. However, after publishing, the Power BI service does not have access to the entire universe of, in this example, R libraries. Microsoft does introduce support for additional R libraries and Python packages in each release update of the Power BI service, but they're not always well publicized.

To see a list of currently supported R packages and Python packages, you'll need to go to Microsoft's official documentation. You can find the list of supported R packages at "Create Visuals by Using R Packages in the Power BI Service" and the list of supported Python packages at "Create Visuals by Using Python Packages in the Power BI Service".

It's also worth knowing that "Data source settings" for any R data source must be set to Public, and all other steps in a Power Query editor query must also be set to Public. This can be done by choosing File → Options and settings → Data source settings, then choosing the Edit Permissions button with the relevant data source selected. At the time of writing, Python also has a known limitation: you cannot use Python scripts in reports using the Enhanced Metadata feature.

Enabling R and Python for Power BI

The first thing we must do to use R and/or Python in Power BI is to make sure Power BI knows where the relevant home directories are so that it can call those libraries and packages when they're used. To do that, go to File → Options and settings → Options. That will bring you to the full options/settings list for Power BI Desktop.

Several advanced settings are available here, but what we care about are the R scripting and Python scripting settings. Power BI Desktop does a good job of identifying the home directories for these languages when they're installed. However, if they are not autodetected, you'll have to navigate to them inside the Options window. Note that in both of these settings, a hyperlink will take you to the relevant Microsoft documentation for installing R or Python. Once that is done, you're good to go.

R and Python in Power Query

First, I'm going to look at using R and Python scripts to modify data, and then we can discuss R and Python as their own data sources.

R and Python scripts can be found in the Transform section of the ribbon in the Scripts subsection, which is farthest to the right. Clicking either button will bring up a dialog box where you can put your relevant R or Python script.

Make a mental note that an R or Python script goes into the query's applied steps in order, like everything else. This has the advantage of being able to be done at any point until the R or Python script is called, and both R and Python will support calling the data in the step before that transformation as "dataset," for the purpose of referencing the data in your relevant script.

R and Python also do not need to be terminal transformations. You can have Power Query transformations occur both before and after an R or Python script is called. This can be helpful when you might find a very specific data transformation difficult to do in the Power Query UI, or when a transformation would require some complicated M.

You can also use R or Python scripts as their own data sources. If you choose "Get data" and search by "script," you'll see both R and Python script options. In scenarios like this, where you may already have a well-defined script that you have developed from the relevant language's IDE, you can just copy and paste that script into the dialog box that appears. This will call the R or Python script that will probably contain a link to some source data, run that script, and display the results as a completely new query. Those results are then fully modifiable in the same way as any other Power Query table in any future transformation steps, and that can include, again, a script transformation that uses either or both languages in sequence.

R and Python Visuals

R and Python visuals are also supported in Power BI Desktop, assuming you have the relevant languages installed. The first time you select Visual from the Visualizations pane, you will get a warning dialog asking if you want to enable script visuals.

Script visuals give you the ability to make custom visuals utilizing either language in a way that can really expand your analysis beyond the visualizations that might normally be available in Power BI—and even those that could be available through the Power BI custom visual options.

These visuals are also fully interactive with other visuals on the report page and slicers. Other cross-filtering from other visuals can, in fact, modify R and Python visuals as well. When you first open an R or Python visual, you will be asked to drag fields into the Values area of the Visualization pane. Power BI will use the fields that are dragged into the visual to define what will be the "dataset" for the visual. We can see what that looks like before and after fields are added in Figure 7-19.

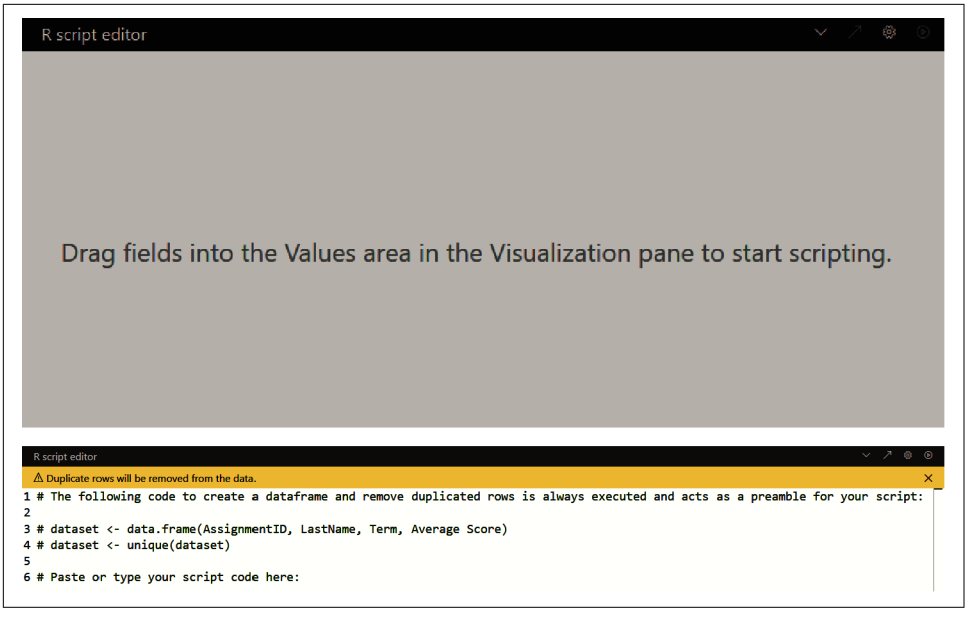


Figure 7-19. Power BI will try to give you a head start on R and Python visuals by defining the "dataset" from your field selections

You'll also notice that, for both R and Python visuals, you will get an alert noting that duplicate rows will be removed from the data. After you have put in your relevant scripts and modified them to reference the "dataset" that you've built in the Power BI interface, you can click the Play button farthest to the right in the script editor bar to run the script. You can click the menu arrow to minimize the window. The arrow pointing up and right will open your relevant language's IDE, so if you need to edit the script, you can do it in a more friendly editor.

Conclusion

In this chapter, we went over some more advanced analytics topics in Power BI Desktop that you can use to extend your analysis. Utilizing AI-powered visuals, what-if analysis, and even R or Python can be tools that help your data consumers get to the information they need quickly and accurately.

This chapter concludes our section on the Power BI Desktop. From here, we will move to an overview of Power BI service and everything that goes into sharing the wonderful reports you'll develop.

CHAPTER 8

Introduction to the Power BI Service

Now that you've completed the previous chapter, you have a beautiful report filled to the brim with insights. You're ready to share your creation with the organization to help make the big decisions...but how? You share by using the Power BI service.

Power BI is more than just the Desktop authorship tool we've been discussing thus far. It's an ecosystem of products, and in this chapter, you'll find out about the second part of that ecosystem, the Power BI service. The *Power BI service* is the software-as-a-service (SaaS) portion of the Power BI infrastructure that allows end users to share reports with one another, manage access to workspaces, create reusable data elements for other report users in the forms of dataflows, and create collections of report elements for broad distribution in the form of apps.

In this chapter, you'll learn the basics of the Power BI service. How do you log in? How do you navigate around? What's a workspace? Why do workspaces matter? How do you use the Power BI service to share reports? There's much to unpack, so let's get started!

The Basics of the Service: What You Need to Know

Initially, you can log into the service and enter your Microsoft account (or employer account) credentials. Everybody can access the Power BI service with a "free" license. That free license gives you access to "My workspace." In this personal development space, where you can publish reports and review how they look and behave in the service.

After providing your credentials, you'll be taken to a home page that looks something like Figure 8-1.

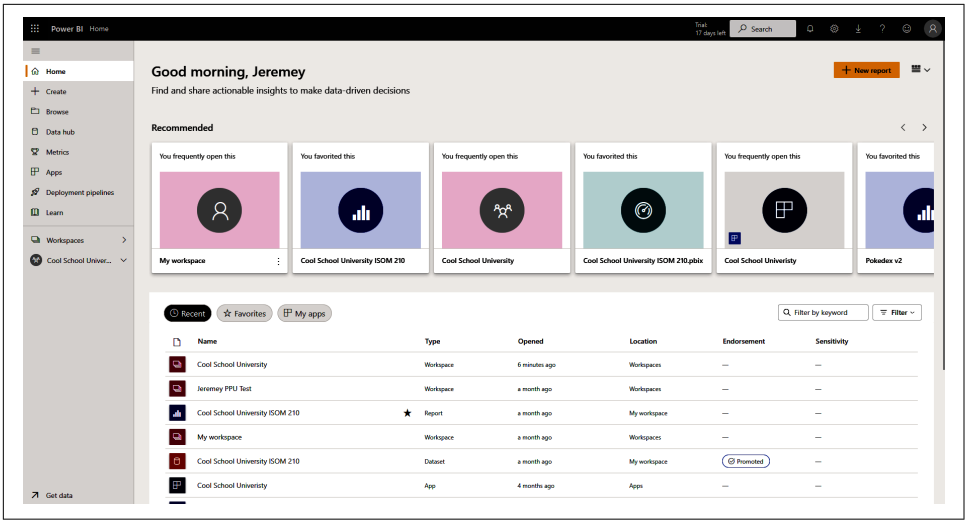


Figure 8-1. The Power BI service home page

You'll notice a ton going on here. There's a navigation bar on the left, a Favorites and Frequents listing in the middle, and underneath that, a link to recent elements and apps you have access to (we'll discuss the context for what apps appear here a little later). At the top right in the darker bar, several options enable searching for objects you have access to, creating new reports, and settings controls. This first look is also the same look you'd get if you were to click the Home link that's at the top of the menu on the left side.

The good news is that everything you really need to access at first can be obtained from that left-hand menu, called the Navigation menu, so let's walk through those areas and get an idea of the most important objects there.

The Navigation Menu

Figure 8-2 shows the Navigation menu up close. In the dark bar at the top, you'll see what looks like a nine-dot box, the Power BI title, and another link to Home.

The nine-dot box is quite convenient. If you click it, a shortcut list will appear to take you to other Microsoft services you might have access to. For example, you can go to Outlook, OneDrive, Word, Excel, PowerPoint, OneNote, SharePoint, Teams, Sway, and other Microsoft products you might have under your Office 365 license. This can be helpful when you're working in Power BI and just need to quickly get to another part of your Office suite. It can also be convenient when you're building something like a dataflow and need to reference where an object might be that you want to pull in from SharePoint or OneDrive for Business.

That Home link will do the exact same thing as clicking the Home button that's shown below, next to the house icon (in the lighter part of the menu). Remember, Microsoft wants to give you a million ways to do something, even if they're literally pixels apart.

Directly beneath that nine-dot box in the Navigation menu, you'll see a hamburger menu icon that will minimize the Navigation menu or expand it, based on your viewing needs. When it's condensed, you can still see the icons of the Navigation menu and use them to get around. When expanded, you'll get explanations of what those icons mean. Personally, as someone who really struggles with remembering icons, even when they're as relevant as Microsoft has made them here in the Navigation menu, I will just leave the menu expanded. I condense it only when I'm looking at a report page and need some extra space. With that in mind, let's dive into the Navigation menu and look through these options, starting with Home.

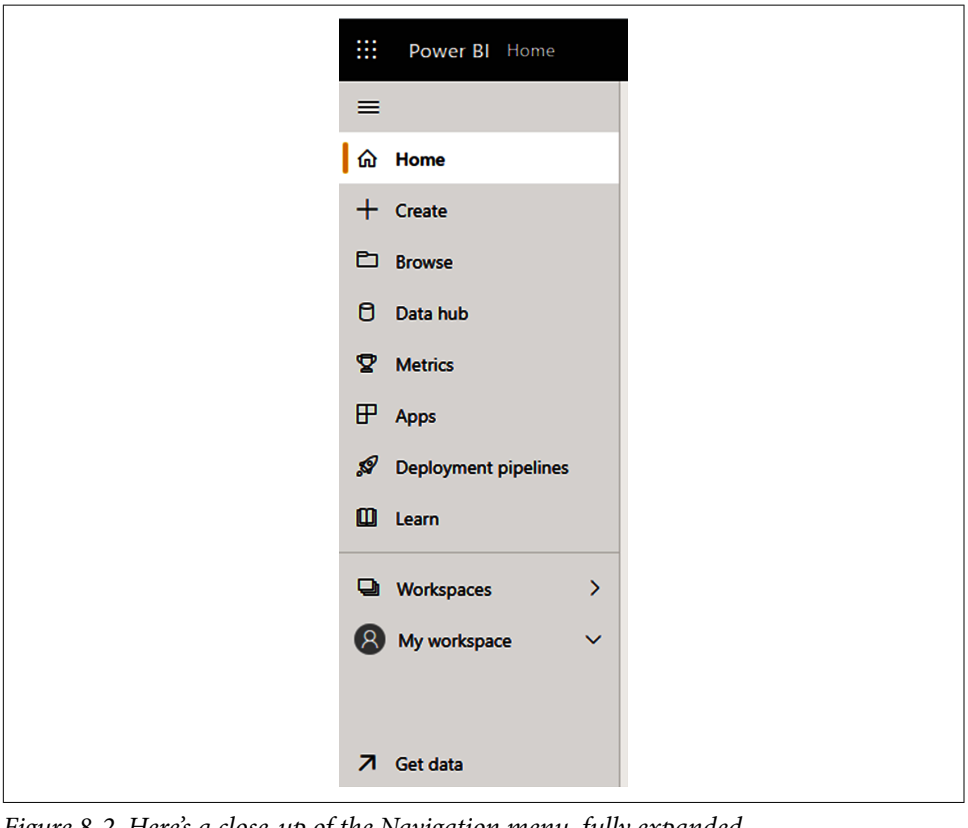


Figure 8-2. Here's a close-up of the Navigation menu, fully expanded

Home and Browse

The Home section contains quick links to objects in the Power BI service that are recommended to you, objects that you've accessed recently, objects that you've favorited, and apps that you have access to.

As shown in Figure 8-3, at the very top of this page is a nice greeting and a button (called "New report") to create a new report, which will take me to the same section as the Create link on the Navigation menu. Next to that is a combination ellipses and hamburger menu icon that will allow me to choose either the simplified or expanded layout. The expanded layout doesn't really look much different, except it has some useful learning links at the bottom of the page. Everything you see in this section uses the simplified page layout.

Along the top, you can see a list called Recommended, containing recommended objects. These can be workspaces, reports, datasets, apps, pretty much anything. Also included in this list can be data elements that are promoted by your organization, such as promoted or certified datasets and apps to which you might have access.

Below the recommended objects is a collection of recently accessed elements, favorited elements, and apps. I can toggle between the views by simply clicking the relevant button. I can also search the lists by keyword, and the Filter button lets me choose specific types of objects to look for, things I've opened in a given time frame, or I can filter by endorsement status.

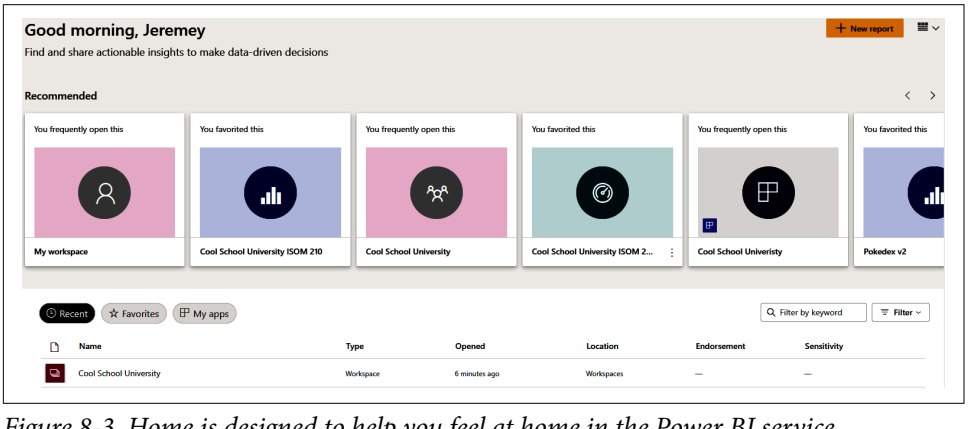


Figure 8-3. Home is designed to help you feel at home in the Power BI service

This layout is very similar to what you might see, for example, in a SharePoint list. I can see the names of the content I've favorited. I can see the type of element, whether it's a report or a dashboard, the owner of that element, its endorsement status, and any sensitivity labels on that element.

In the Favorites list, you can see where Power BI still shows those items as starred, even though that feels a bit redundant. Well, that same functionality exists in the same unmarked column in Recent. If any recent objects are also in your Favorites list, you will see those elements starred.

The Browse section of the Navigation menu is, in context to the Home section, pretty redundant. It shows you a list of objects that you have access to, and they are broken into subpages for Recent, Favorites, and then specifically items that have been shared with you. As you can see in Figure 8-4, Microsoft is nothing if not persistent in giving you too many ways to do the same darn thing.

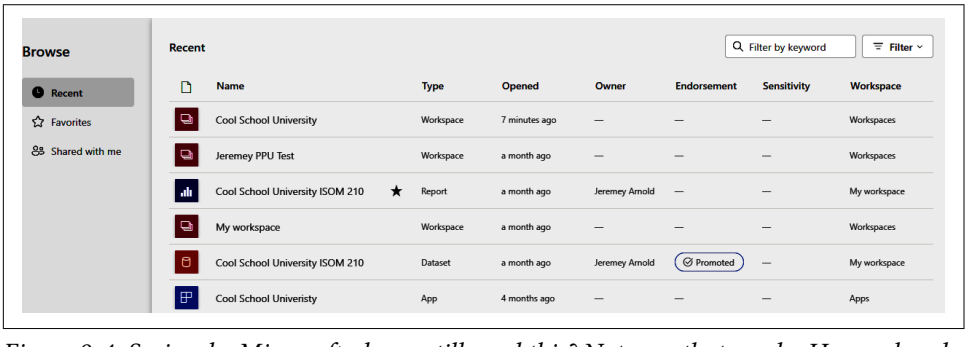


Figure 8-4. Seriously, Microsoft, do we still need this? Not sure that we do. Home already looks so nice!

Create

Power BI offers a way to build reports in the service itself. Honestly, it's one of those things that exists, but the use cases for doing it in the web versus downloading Power BI Desktop for free and using that to generate a report aren't very common.

If you can use Power BI Desktop for authorship, you should. With that in mind, the Create tab will give you two options. You can manually enter or paste data into a table sheet, or you can pick another published dataset in that workspace and create another report based off that dataset. We can see what this interface looks like in Figure 8-5.

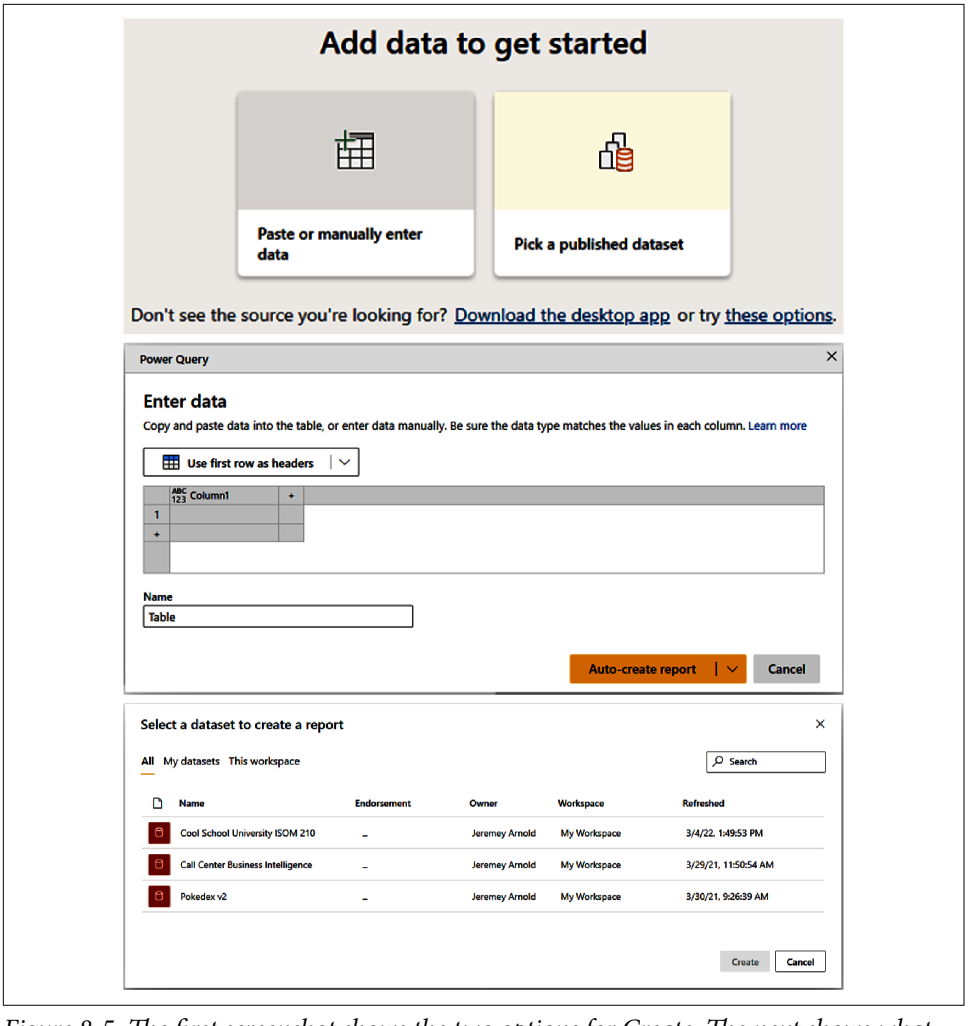


Figure 8-5. The first screenshot shows the two options for Create. The next shows what happens if you click the manual data entry option. The third shows what happens if you select "Pick a published dataset" to build a report on.

Data Hub

In the "Data hub section," you can see a list of recommended datasets, a list of all the datasets you have access to, and the specific list of datasets that you have authored. As of the time of writing, a new feature in preview called Datamarts is also visible here if you have Premium Per User or Premium Per Capacity licensing.

You'll note that Microsoft makes some hyperlinks available in this view that will send you to their documentation if you have questions on specific elements on this page regarding dataset discovery and explaining what exactly a dataset is.

With each dataset, a vertical three-dot selection brings up a menu that gives you an expanded list of options for that dataset. We can see what that looks like in Figure 8-6.

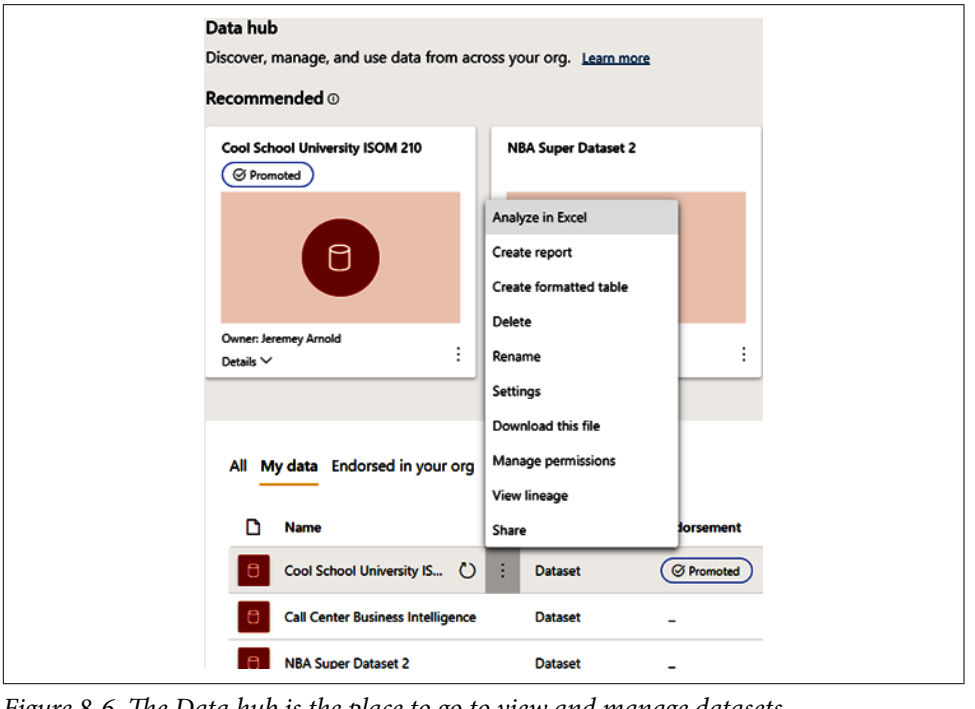


Figure 8-6. The Data hub is the place to go to view and manage datasets

You can do several things with a dataset from this menu. I'm going to approach them a little bit out of order, putting settings last because that's really its own section.

The first option we see is Analyze in Excel. The first time you do this, Power BI will ask you to install an update to Excel to read the file format. Once that is done, Power BI will generate an Excel worksheet that will already have a live connection setup to the dataset it originated from.

This is very similar to connecting to a Power BI dataset to create a report from Power BI Desktop. The data is opened in a pivot-table format, and you can put fields into rows and columns and insert measures into the Values section, just as you would any other pivot table. This is helpful to users who may know how to leverage the data model you have created, but aren't yet comfortable with Power BI Desktop (in which case, might I suggest a copy of this book for them?). Or maybe the user just needs some quick pivot-table analysis in Excel. The advantage of this function, from an Excel perspective, is that the only data stored locally is the data that appears in the cell itself—because the core of that data is in the cloud.

"Create report" will take you to the web report authoring experience, where you will see all the tables and measures in the dataset. You can use them to build a report with as many pages as you need, and that report will exist in the Power BI service. One word of caution: the web authoring experience does not allow you to edit a dataset. If you are missing a measure or need to add a table or something, you can't really do that from the web authoring experience. You'd have to open the original dataset, add your relevant data elements, and then republish that dataset back to the service.

"Create paginated report" will create a Report Definition Language (RDL) file that will have all the necessary connection information to create a pixel-perfect report, a la SQL Server Reporting Services, to be hosted in the Power BI service. Paginated reports for the Power BI service are not authored in Power BI Desktop but are authored in Power BI Report Builder, which is a different software package. You will see this option available in only Premium Per User or Premium Per Capacity workspaces.

"Manage permissions" will allow you to see all the objects for which you have the ability to add or remove users. You can see who has direct access to a report, dataset, or workbook. You can add users manually from this interface and see which users have requested access but whose requests are still pending. See an example in Figure 8-7.

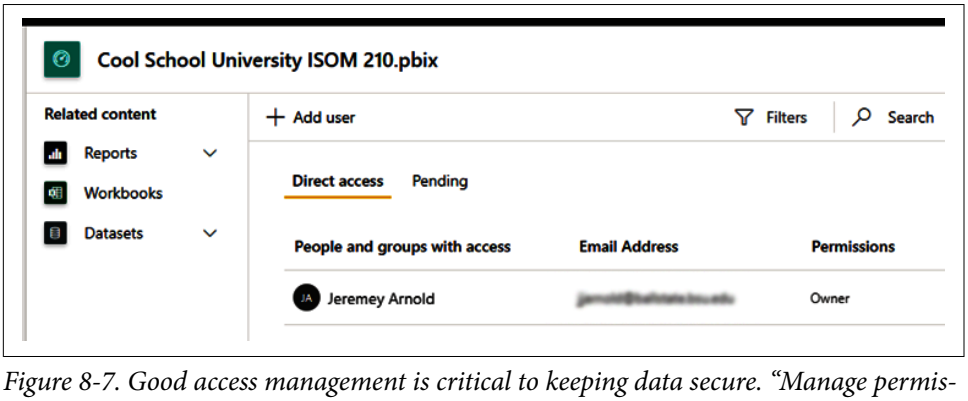


Figure 8-7. Good access management is critical to keeping data secure. "Manage permissions" lets you see who has access, add users, and see who has requested access but is still waiting.

"Chat in Teams" will create a link that is shareable to a person, group, or channel in Microsoft Teams. Users can then access the dataset via Teams, bringing them to the appropriate location in the Power BI service. You can also share reports and report pages in Teams. Microsoft Teams can be a great way to embed analytics into your organization in a framework that users will already be familiar with. This opens the door to using Teams as a channel to discuss the results of a given report or dataset and gives you, the author, the ability to provide quick insights, address questions, or make improvements based on user feedback.

"View lineage" opens a new browser window where you can see all the data elements inside a given workspace and their dependencies. For instance, if I had a dataset that utilized a SQL Server instance, that SQL Server would show up before the dataset and then all the reports on that dataset would follow, and then any dashboards that were created from report elements would also be shown.

In my personal workspace with the datasets I currently have, they're all based on flat-file data, so there's no element before the dataset. We can see what that lineage view looks like in Figure 8-8.

The other thing you can do from this view is create a number of other elements by using the New button. This will allow you to use web-based design tools to create new elements directly into a workspace or to upload PBIX files directly into a workspace. This can be used for reports and paginated reports, scorecards, dataflows, and even datasets. If you have an opportunity to use Power BI Desktop instead of the web authorship tool, I would always recommend doing so. For things like dataflows and scorecards, though, this is where you'll create those.

In my opinion, this is a strange location for this functionality to live, especially for items better suited to Power BI Desktop, but it's the single most cohesive place that these options exist from a web authorship perspective. At the time of writing, Microsoft does plan to provide a web authorship tool for paginated reports, and at that time, this option may push you to the web authoring for said future paginated report.

Clicking any dataset will show you a list of tables and columns in that dataset. "View lineage" will allow you to see how data elements flow from one item to another, showing you potential impacts that would result from making a change to a dataset, even beyond your single workspace!

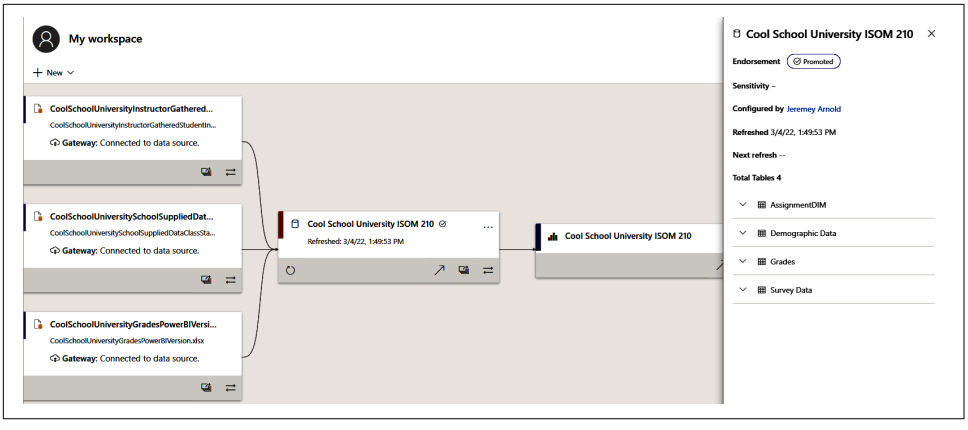


Figure 8-8. "View lineage" will help you see what data came from where and what is using it now

Settings

The Settings selection on a dataset will take you to a properties page preselected to the Datasets portion of the Navigation menu at the top of the page. You'll see a list of all the datasets in the workspace on the left. On the right, you'll see several properties options for the currently selected dataset that are very important to the management of your dataset(s). Figure 8-9 gives us a look at this page.

The first option, which is the "View dataset" hyperlink, takes us to the dataset's workspace page, which is an area we'll discuss when we get to Workspace navigation. The second option, "Refresh history," opens a pop-up inside the page that will show you the dataset's refresh history, indicating whether it was successful; and if there was a failure, it provides a reason for that failure. Some failure messages are helpful. Some are not. It's still a Microsoft product, and Microsoft's error messages can be hit-or-miss in terms of their helpfulness; they're notoriously unfriendly when it comes to troubleshooting in Windows. Next, we can create or update the dataset's description.

For data sources that are not in the cloud or more specifically in Azure, you will most likely need a data gateway installed to facilitate that dataset's refresh. For example, let's say I have my Power BI dataset refreshing from a couple of Excel files on my company's network drive. The Power BI service doesn't have access to my network drive.

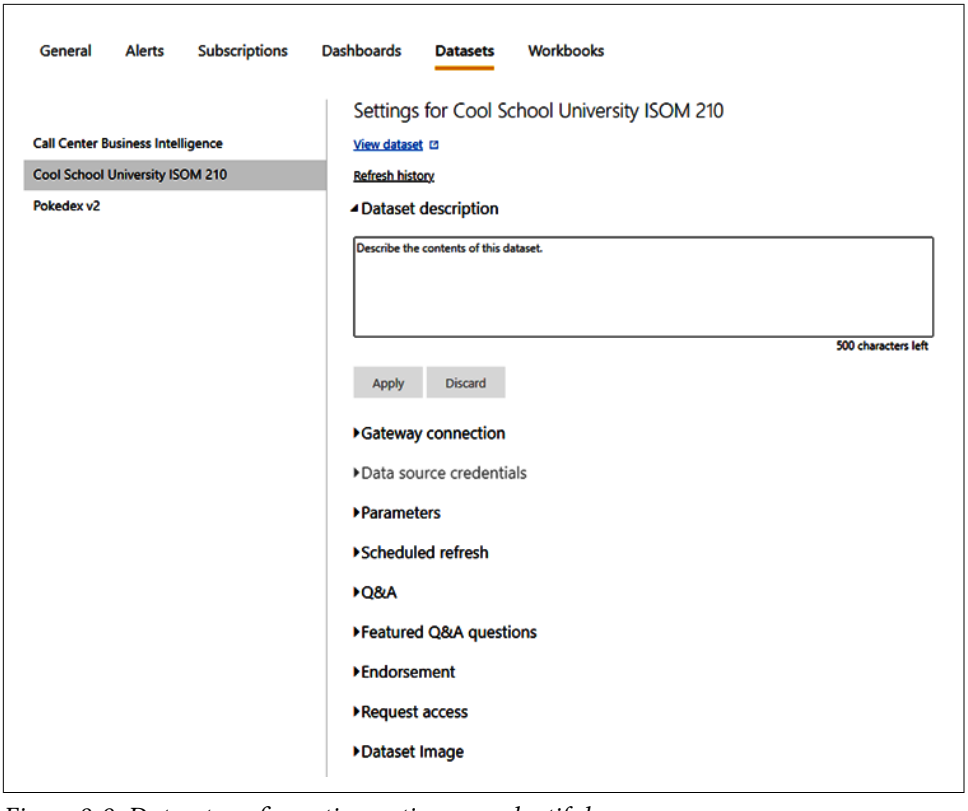


Figure 8-9. Dataset configuration options are plentiful

A data gateway acts somewhat like a virtual private network between the Power BI service and the locations allowed by the data gateway to facilitate Power BI being able to get the data for refreshes. Going over each data source and whether it requires a gateway would take a long time, and admittedly Microsoft would be the best and most up-to-date source for that.

If you need to check whether a data source requires a data gateway, go to "Power BI Data Sources". Gateways will typically be managed by your IT or security team, so if your organization is already using an enterprise gateway, reach out to the relevant stakeholders to see if you need to get a certain data source added to it.

Microsoft does make a personal version of the data gateway available for download. This version runs on your local machine, and assuming your machine is on and the gateway is running at the time of refresh, that will also work. Take note that all R and Python data sources and transformations require the use of a personal data gateway, not an enterprise one. This is because the Power BI service needs to run the relevant R and Python scripts from your machine since, presumably, your machine has all the necessary packages and libraries.

Next, although grayed out in Figure 8-9, "Data source credentials" would be where you configure your credentials required to access the data. In this case, let's say you were pulling data from an on-premises SQL Server and that SQL Server instance was configured in your data gateway; you would use this section to provide the relevant credentials so that Power BI would know what credentials to use to pass through the gateway and onto the original data source.

A Power BI file when uploaded will generally keep the credentials in place that were used in Desktop, but that doesn't mean that's what you want. For example, you may want to transfer the credentials used for refresh to a service account that has been set up. Or you may need to change credentials when someone has left the organization or for myriads of other reasons. Power BI will do a good job of telling you if something needs to be checked here, so if you are having refresh issues, it never hurts to double-check your data source credentials.

Parameters are straightforward. If you have parameters in your dataset, you can configure those parameter values here. One use case is that I like to parameterize my server names and database names. If something should happen and I need to quickly repoint my Power BI dataset to a backup server, or the name of a database gets changed, I can come into these settings and basically hot swap those connections, refresh the data, and move on with my day. All your Power Query parameters for a given dataset can be managed and have their values updated here. When those parameter values are updated, it will push a refresh to accommodate the newly updated data state.

"Scheduled refresh" is where the real magic happens in the Power BI service for me. This is where you will configure what time and how often your Power BI dataset will refresh against its source data. You can set up to 8 scheduled refreshes in nonpremium environments and up to 48 refreshes a day in premium capacity environments. You set your time zone, whether it's daily or on specific days of the week, and you save those settings. Just because you set up refresh doesn't mean it will work. Make sure you've successfully refreshed your dataset in Power BI Desktop first and make sure all your settings are correct for the dataset in service before setting up a scheduled refresh.

We talked about Q&A quite extensively in Chapter 7. Q&A allows you to turn on or off Q&A functionality on the dataset. It also allows you to share your synonyms with everyone in the organization. Likewise, "Featured Q&A questions" allows you to put featured questions in front of your audience when they view reports based on this dataset.

The Endorsement feature allows you to identify a dataset and highlight it across the organization. Microsoft makes three levels of endorsement available: None, Promoted, and Certified. None is the default setting. When a dataset has no endorsement, it will show up in search results, but that's it. When a dataset is promoted, it will show

up in search results, and that result will be promoted in the search context. You can also choose to make that dataset discoverable, which means users who don't have access to that dataset will be able to discover it and request access themselves. Finally, a dataset can be certified. A *certified dataset* is one that has been reviewed, and this is a sort of "source of truth" label. By default, an individual doesn't set a dataset to Certified status. That is done by processes in your organization. Each organization will have its own criteria for when a dataset can be labeled Certified.

"Request access" will let you determine how users can request permissions to access content related to the selected dataset. You can have a request emailed to the dataset owner, or you can have an automatic response providing a set of instructions for the follower to get access.

Finally, Dataset Image allows you to choose a nice picture that will show off the dataset in the places where it is discoverable. This image will be available and represent this dataset everywhere in your organization where a dataset can be discovered.

Metrics

Metrics are a relatively new feature to the Power BI service. They allow you to create trackable KPIs or goal metrics by using data in the Power BI service. You create scorecards that show the performance of a given metric over a given time period.

Metrics are always going to be connected to a specific user or business case, so I don't want to get too into the weeds with this. If you think you might have a use case for metrics, I recommend just trying it out. Microsoft recently enabled working with metrics in your personal workspace, as well, so try to work on creating some goals for yourself before expanding into this functionality for other parts of the organization.

Apps

An *app*, in the context of the Power BI service, is a collection of packaged content that can be distributed to a broader audience. Apps are created inside a workspace and then, when they're ready, can be published to a collection of individuals, a Microsoft 365 group, or an entire organization. Apps can also have different permissions than those in a workspace, making user management a bit easier since the app's permissions are managed in one location.

Users can make apps from a workspace. Many apps also are available for you to use, called *template apps*. These apps are collections that others have put together and shared so that you and others in your organization can use them.

In Figure 8-10, we can see the initial Apps view, which will first appear empty. We can then add either a template app or an organizational app and view the content from that application. We can use the yellow "Get apps" button in either location to access the list of available apps that we can bring into our workspace. In my example, I've

created an RLS Test application that will be an organizational app, and the others are template apps.

We will walk through the installation of the template app using Microsoft's COVID-19 US Tracking Report template app. Notice that in the "All apps" section, organizational apps are always listed first before template apps, but in both groupings they're not really organized in a meaningful way to me. You'll note there is a place where you can choose to select only endorsed apps. This can be helpful if you're looking for a collection of data that has been endorsed inside your organization quickly.

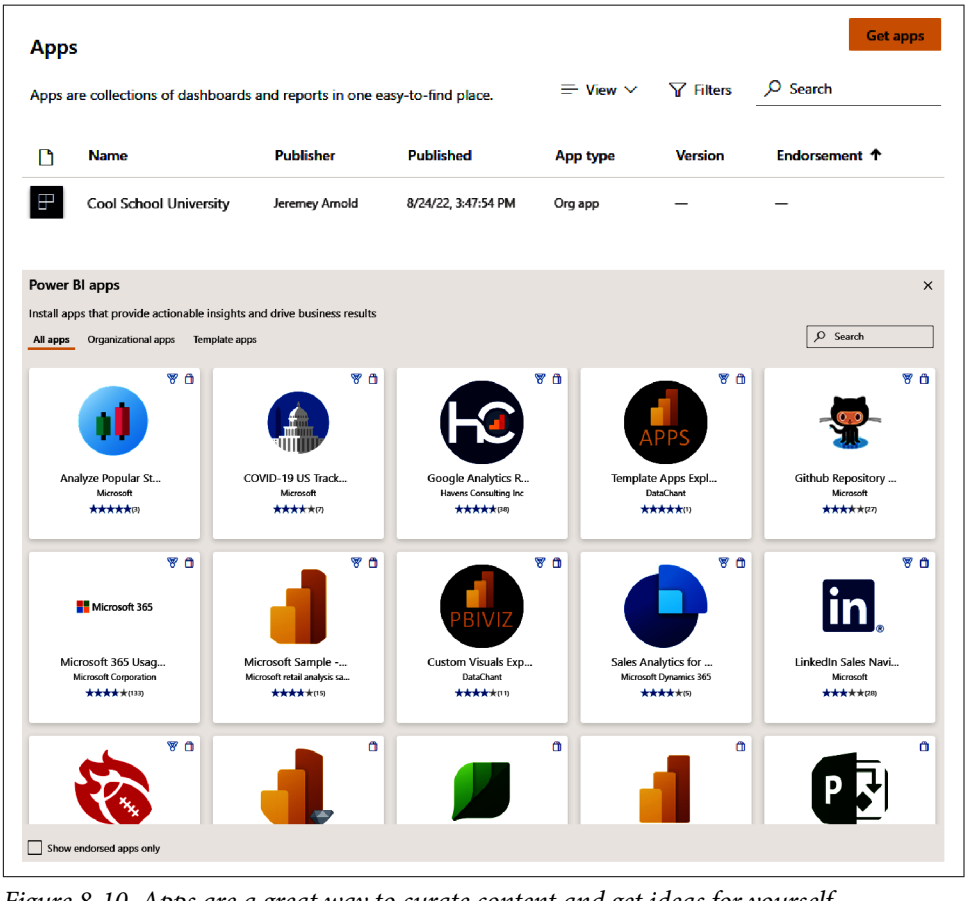


Figure 8-10. Apps are a great way to curate content and get ideas for yourself

Installing the application is as easy as clicking the appropriate box and being taken to the relevant page in AppSource (Microsoft's application and custom visual portal). Some apps may have licensing involved, so you'll want to check that in the description, but in most cases, there will be a big Get It Now button.

In the case of Microsoft's template apps, they will have a pop-up before installation that will ask you for some information for marketing purposes. This does not have to be associated with the same information as your Power BI account itself, so if you're uncomfortable, feel free to use a throwaway email.

Once the app or apps are installed, they'll show up here. Accessing them is as simple as clicking the link under its name. Each app will have a list of information, including the publisher, the date it was published, the type of app, the version number if relevant, and its endorsement status. Once an app is installed, you can look at the reports that came alongside that app. I enjoy looking at some of the template apps for design ideas for my own personal work, so don't be afraid to borrow clever ideas!

Deployment Pipelines

Deployment pipelines are a Premium Per Capacity–only feature that allows you to designate Development, Test, and Production environments for a given Power BI workspace. This can be a tool used by developers to help with issues like version control and user testing.

Workspaces can either be created or assigned to support a pipeline, but as mentioned before, all such workspaces must be in Premium Per Capacity.

Deployment pipelines are typically managed by either workspace- or tenant-level administrators, or whichever group inside an organization manages the Premium Per Capacity tenant. If you happen to be on a Premium Per Capacity tenant and have a need or desire to establish a pipeline, work with your relevant administrator to get that put together or ask to have work added to an existing pipeline.

Learn

Learn is a small learning portal where Microsoft gives you links to training resources, Power BI documentation, and some sample reports to get you started. In Learn, you can also join the larger Power BI and Power Platform communities.

I cannot speak highly enough of the Power BI community. Many great users take time to create resources, answer questions on forums, promote new ideas for the product, and create events for practitioners to share what they've learned so others can learn from them as well.

Publishing Your Work

Now that we've discussed where and how you'll navigate the Power BI service, we need to get content in here so that others can learn from our data! We've dealt with the first half of that Navigation menu, but the things in there really matter only if we get content into our workspaces.

To publish your work, you will take your PBIX file from Power BI Desktop and can either use the Publish button in Power BI Desktop from the Home ribbon or upload a PBIX file to the Power BI service from the Navigation menu. You may have seen the "Get data" button at the bottom of Figure 8-2. When we click that button, we see the page in Figure 8-11.

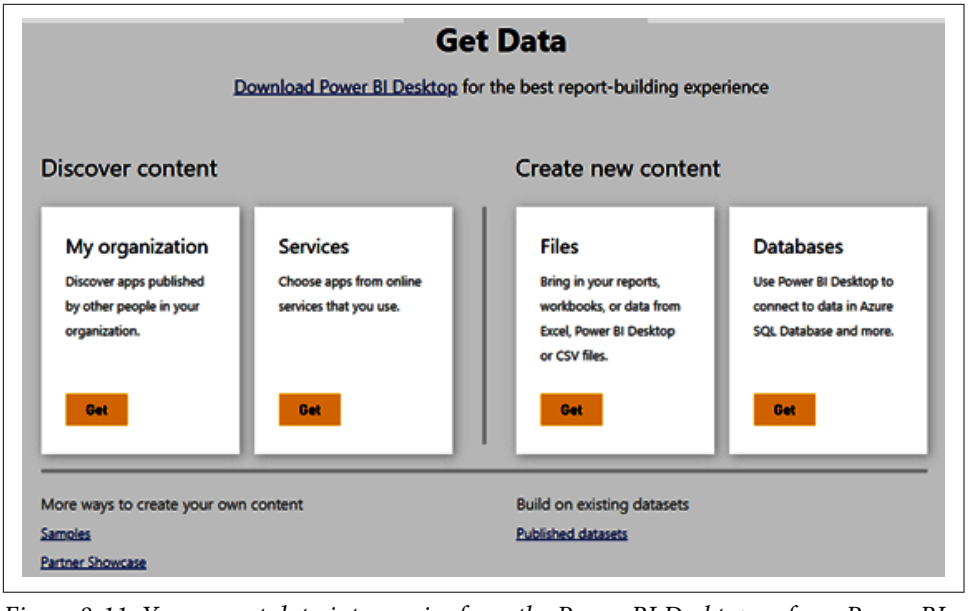


Figure 8-11. You can get data into service from the Power BI Desktop or from Power BI service

In Power BI Desktop, you will see a menu that shows a list of workspaces you have access to, and you'll choose which workspace to publish the report in.

In the service, it's more of a pull-up function than a push-out function. You'll upload the PBIX file from the workspace that you wish to add the dataset and/or report to. On the left, we see the Discover content functions that will allow you to see your organizational apps and other template apps that can be plugged right into the workspace.

On the right side is a Files option that you can use to upload your PBIX file.

The Databases & More option allows you to work on creating a dataset from a connection to Azure SQL, Azure SQL Data Warehouse (which is now Azure Synapse, and I don't know why Microsoft hasn't updated this; my guess is because almost no one uses this), SQL Server Analysis Services, or Spark on Azure. In Figure 8-12, we'll see what service looks like when I select File, because that's really what you'll use this option for 99% of the time.

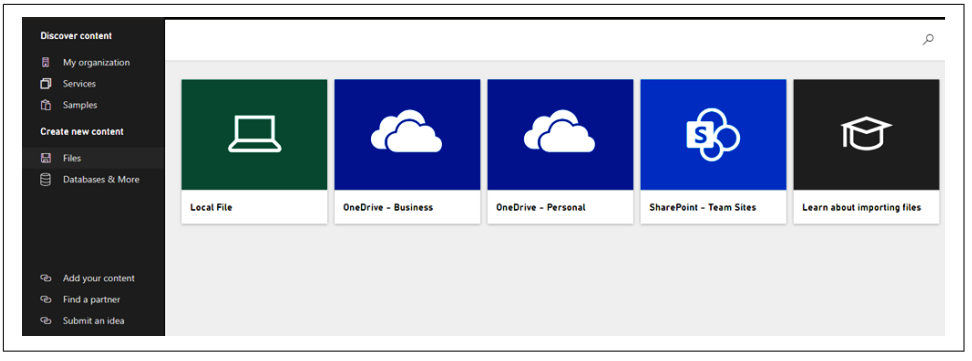


Figure 8-12. Uploading files? This is where you'll be.

From here you'll see a couple of options. Local File, in this case, means your PBIX file that you generated from Power BI Desktop. The OneDrive links allow you to create a dataset in the Power BI service that's connected to either your OneDrive for Business or OneDrive personal accounts.

From a refresh perspective, these sources are convenient because they do not require a data gateway to refresh. However, I still suggest building your report in Power BI Desktop using OneDrive sources, as opposed to doing that here. Likewise, the same is true for the SharePoint option. Note that in scenarios where you might be using OneDrive or SharePoint for version-control purposes, you can still upload PBIX files as well via those channels, and that can be a convenient use case for these nonlocal file options.

When Local File is selected, you'll see your classic Windows Explorer window open. Navigate to the appropriate file and upload it. After that, the dataset will appear in the workspace navigation area for the selected workspace (which you can see under Workspaces if you look back at Figure 8-2).

One other point: whenever a dataset is uploaded to a workspace for the first time, it also will automatically create a blank dashboard with the same name as the dataset. So you see that getting our Power BI work into a workspace is pretty simple, but what actually is a workspace and why do they matter?

What Is a Workspace?

A *workspace* is simply a place where Power BI data assets are held. Datasets go into a workspace, reports go into a workspace, dashboards go into a workspace. A workspace can be used as a place to send people to get content, and it can also be used as the basis for an app that you would send users to for content. When we publish datasets and their reports to the service, we are publishing them to these workspaces.

My Workspace

Everyone who logs into the Power BI service will have a personalized "free" workspace. You need to know about some important limitations of this workspace.

First, while you can share content that is in your personal workspace, both you and the users you share with are required to have at least Power BI Pro licenses.

Second, it's generally considered a best practice to not share content out of your personal workspace permanently, since access to that workspace can be an issue if you should ever leave the organization. Also, you cannot make an app out of content in your personal workspace. Sorry, you can't use this to get around Microsoft's licensing.

The other thing you cannot do is create a Power BI dataflow. The best use for one's personal workspace, in my opinion, is to view it as a personal test environment. You build your report, you think it looks good in Power BI Desktop, you publish it to the service, and you look at it there and make sure all the things are behaving the way you want. Does the report still look good with different screen dimensions? Does the navigation of the report flow the way you want when you don't have Power BI Desktop's extra functionality? These are questions that you can answer from your personal workspace. You can also test your scheduled refresh in this workspace, which can be helpful before moving that dataset into a more permanent home in a shared workspace.

Shared Capacity Workspaces

In a *shared capacity workspace,* people who have Power BI Pro or Power BI Premium Per User licenses can share Power BI data elements with users who also have access to that workspace.

Creating a workspace is a very simple process. When you click the Workspaces button in the Navigation menu, a pane will appear to the right of the Navigation menu, showing you the list of workspaces you have access to.

In this case, I've created two workspaces to go along with "My workspace." One is a Premium Per User workspace, and one is a normal shared capacity workspace.

Some features work only in a Premium Per User workspace. Premium Per Capacity workspaces will have a diamond symbol next to their names, so you can tell if it is a Premium Per User or Premium Per Capacity workspace, as opposed to a typical Pro license shared workspace. Get a quick look at that in Figure 8-13. At the bottom of that list, you'll see the option to create a workspace.

The workspace that is created will show you options in alignment with your licensing. In my example, I'm taking advantage of the 60-day free Premium Per User trial so I can demonstrate the creation of both types of workspaces.

When you go to create a workspace, a pane will appear on the right side of the page. It will have a place to put the workspace name and description, alongside an image to describe the workspace.

The advanced section of this pane also has several settings. We can choose who will be on the contact list for this new workspace, attach a OneDrive location for file hosting for the workspace, and identify which license mode is attached to the workspace (in this case, Pro, Premium Per User, Premium Per Capacity, or Embedded).

Embedded and Premium Per Capacity will be grayed out unless your organization has that licensing in place. Premium Per User will also be grayed out if you don't have a Premium Per User license.

You can choose to identify the workspace as being used for the development of a template app, and you can allow contributors to update any app that comes from this workspace. This can be good when you have multiple people working in development of a given Power BI solution and you want to have more than one person who can deploy the updated app.

It is important to note that Pro and Premium Per User workspaces can be accessed in a licensing hierarchy. For instance, if I have a Pro workspace, users with either Premium Per User or Pro licenses will be able to access it and the data elements inside. However, with a Premium Per User workspace, only people with Premium Per User licenses will be able to access that workspace. This rule does not apply to organizations with Premium Per Capacity licensing, as that allows for an infinite number of readers inside an organization to any workspace made on its premium capacity node.

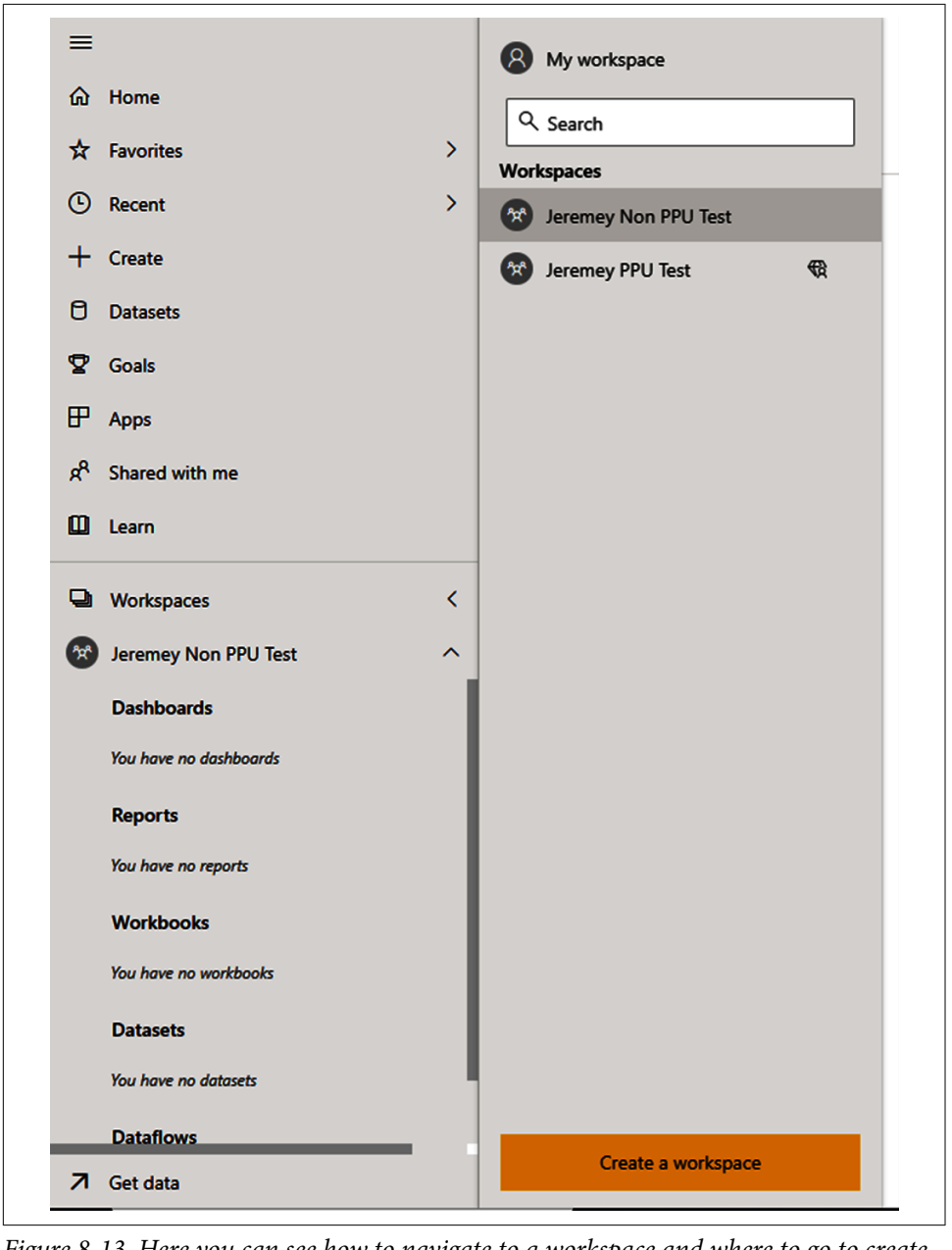


Figure 8-13. Here you can see how to navigate to a workspace and where to go to create a new workspace

Dataflows in Shared Workspaces

With an understanding of what a shared workspace is, we can now discuss the creation of dataflows in a workspace. A *dataflow* is a shared data element that is stored in a workspace that can be called as a data source for report development in Power BI.

Think of this as an ETL process that gets data from somewhere and does some transformations but creates a data element that can be used for further analysis outside a specific model context.

There are two types of dataflows nowadays. There are what I'll call classic dataflows and streaming dataflows. Any workspace that has one type of dataflow cannot have the other type of dataflow in the same workspace. This may change, but as of the time of this writing, streaming dataflows are still in preview.

A *classic dataflow* is a collection of tables that are created in the Power Query in Power BI service. You'll find the option to create a dataflow in the Get Data section. It's a new option under the "Create new content" section that wasn't there when we were looking at that page in my personal workspace. A Power BI dataflow does not have all the same data sources available as Power BI Desktop does. We can see what data is available to Power Query Online in Figure 8-14.

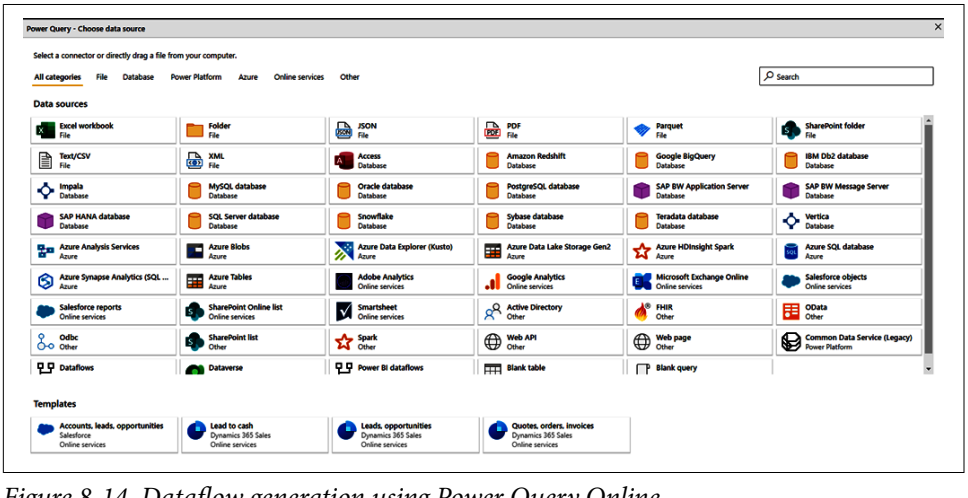


Figure 8-14. Dataflow generation using Power Query Online

It is a pretty good list of common data sources, and Microsoft does add new data sources to Power Query Online from time to time. It can be nice to have a set of transformations or pieces of a data model that aren't locked behind a single dataset from a Power BI Desktop file. And it is nice to have reusable data elements that other people in the organization may be able to leverage for their own analyses. Classic dataflows can be used on Pro licenses.

Streaming dataflows require at least a Premium Per User license. Reports shared from streaming dataflows can be consumed only by users with a Premium Per User license or in a Premium Per Capacity environment.

A streaming dataflow allows Power BI to call either an Azure Event Hub event or an Azure IoT event. It allows you to consume data from either of those sources and do transformations with them in real time so you can do "real-time" reporting on data coming from these scenarios. While this is a cool feature, being limited to Azure Event Hubs and Azure IoT Hub does limit its functionality, and there's a little more setup for this type of event. I personally hope that they expand the number of options before going to general availability and that they make these easier to use.

Putting Your Data in Front of Others

So, while we can put our data into a workspace, we still need to get it in front of people. To do that, we have a couple of options. We can bring users into the workspaces we create. We can create an app. We can link reports to Teams or to SharePoint. We can also embed report elements into a website or into an application if we are using Power BI Embedded. If you want to embed reports into an application, I recommend working with an application developer who will be able to help walk through those technical hurdles. Otherwise, let's get sharing.

Adding Users to a Workspace

The easiest way to get someone to see our data is to invite them into our workspace. Doing that is easy.

In Figure 8-15, we can see the view we get when we select a workspace from our workspaces list. In this case, you see my non-Premium Per User workspace I created. You'll see in the upper-right corner an Access button. Click it, and a pane will appear on the right that allows us to add or remove users and identify the role they play in the workspace.

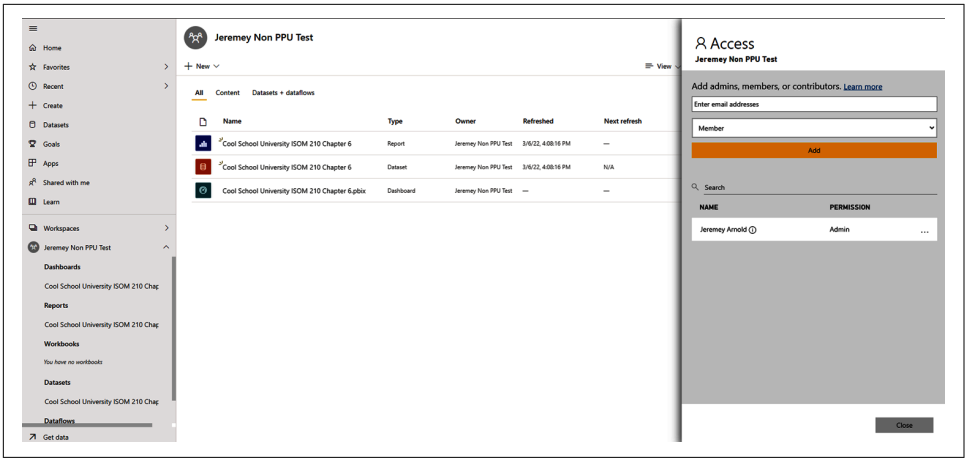


Figure 8-15. Adding additional users is elementary

You will also see an area to add people or groups via email addresses. This is nice because in an organizational setting, Power BI can pretty seamlessly integrate with your active directory instance, letting you search for people in your organization and confirm their email addresses. Click Add, and they're in.

This also works with security groups if you have a group of people you want to add to a workspace in bulk. You can add a user at four levels: Admin, Member, Contributor, and Viewer. Each role has certain levels of permissions, with Admin being the highest and Viewer being the lowest.

If you'd like, you can review what each workspace role can do at "Roles in Workspaces in Power BI". However, my general rule of thumb is that the person who creates the workspace is the administrator. Anyone else who does development in the workspace is a member. Anyone who is there to read reports is a reader. I don't tend to use the Contributor option much, but if you do find a really good use case for that, I'd love to hear about it!

Sharing via a Link or Teams

You can also send out a shareable link to a data element from that data element itself! So, if I look into the shared workspace where I placed my report, I can see a number of options above the page of the report I am currently looking at.

Not everyone who sees the report will have all the options here, but from a sharing perspective, I think it's worth highlighting both Share and Chat in Teams.

Share will create a link that someone can use to view the data element as though they had the viewer role.

Chat in Teams will first, if necessary, have you sign into your account and then create a link to the report page and allow you to share that to a person, group, or channel in Teams. This is useful when you are working on development and want to get feedback or you have really critical KPIs that you want to have an open communication channel on. Chat in Teams is awesome for quick collaboration. We can see what both of these dialog boxes look like in Figure 8-16.

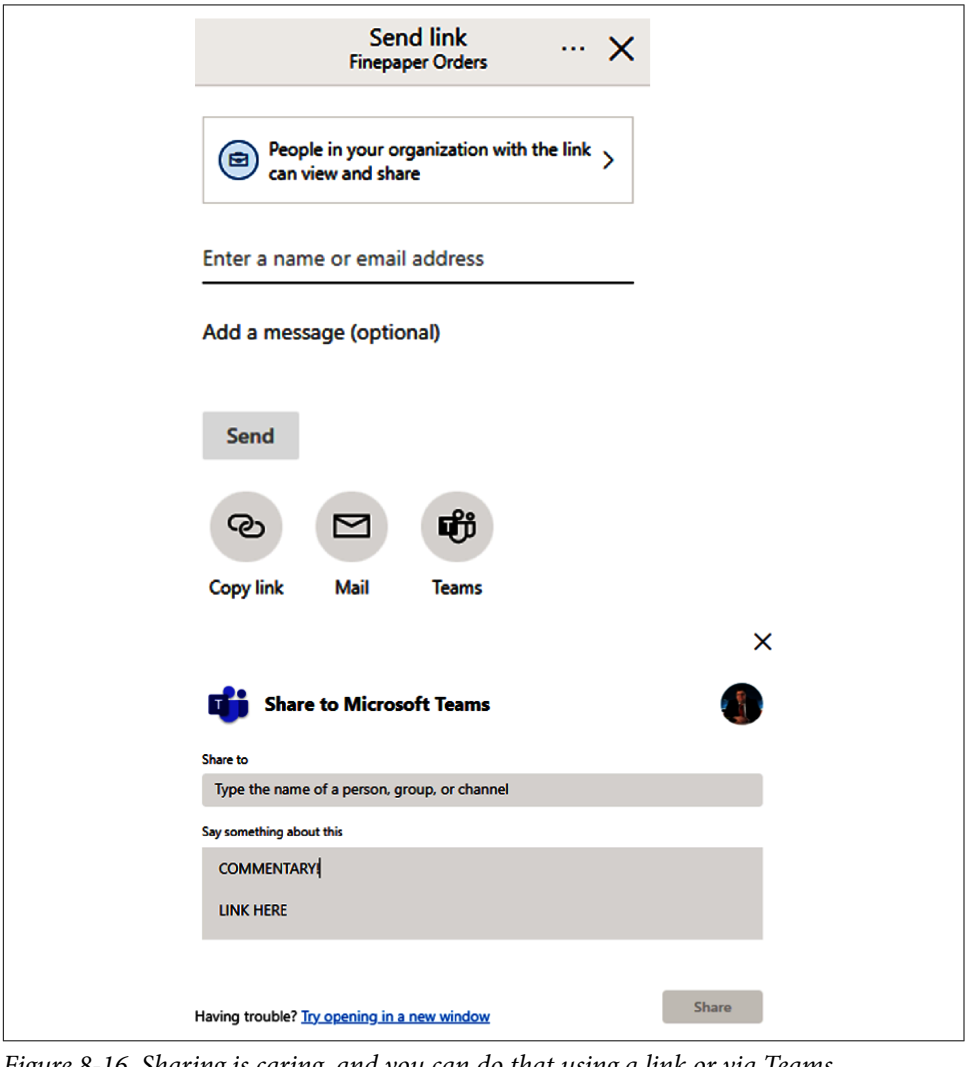


Figure 8-16. Sharing is caring, and you can do that using a link or via Teams

Sharing via SharePoint

If you want to share a data element to a SharePoint page, that's also quite simple. On any given report page, look to the left of Share and Chat in Teams, and click the File option. That drop-down menu has options under "Embed report" for SharePoint online, embedding a report in a website or internal portal, publishing to the web as a public report, or using the Developer Playground for Power BI Embedded testing. We can see this in Figure 8-17.

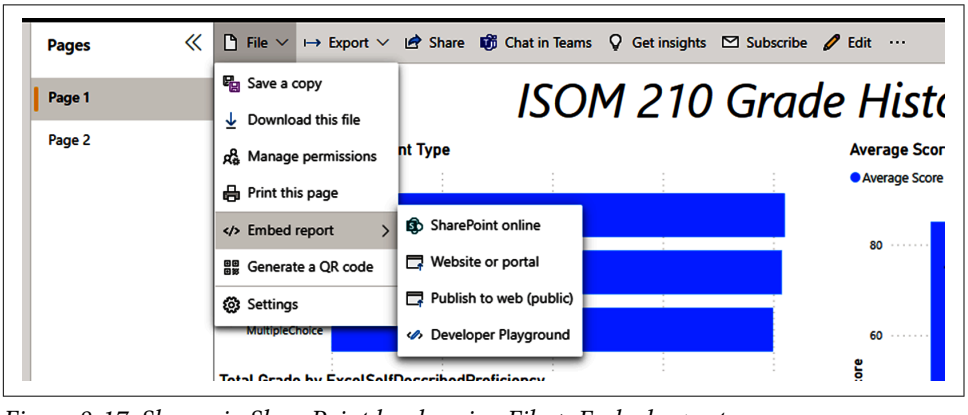


Figure 8-17. Share via SharePoint by choosing File → Embed report

The SharePoint option will give us a website link that we can use to embed in our SharePoint site. This can be done with a Pro license or higher. You'll then go to the relevant SharePoint site you want to add this to and follow a couple of steps.

Note that this feature works only on modern pages in SharePoint. Once we have the link, we can create a new modern site page, select Power BI from the drop-down menu, add the report, provide the copied URL into the Power BI report link, and then publish!

Creating an App

We've discussed apps a couple of times in this chapter, and creating an app is very simple. Select your workspace that has the data elements of your future app. You'll see, on the far side of the workspace items, a toggle that lets you choose whether it should be included in the app.

Datasets are not shared in apps; only reports and dashboards that use that dataset will be included. The app still goes back to the dataset for its information, but it hides the dataset itself from app users. In the upper-right corner, once you have your items selected, click "Create app," and that will take you to a page like that in Figure 8-18.

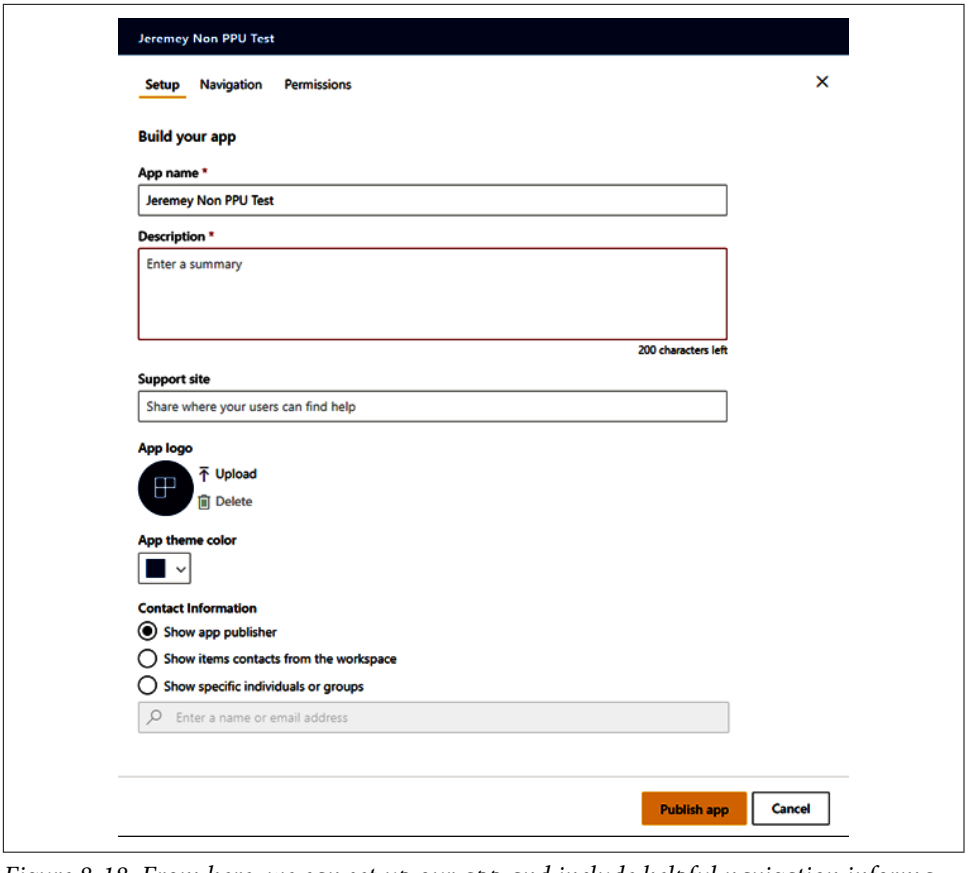


Figure 8-18. From here, we can set up our app and include helpful navigation information and control access

There are controls to choose the app's name, enter a description, create a link to a site where end users can get help or read documentation, determine app navigation settings (I really like the default navigation of the navigation builder so I tend to leave this alone), and set permissions around who can access the app. Is it the entire organization? Is it a specific group of users? Who can access the underlying datasets with build permissions? Can they make copies of reports? Can users share, and finally, should this app be installed automatically? You might want to discuss some of these settings with someone responsible for data governance in your organization, as they may have policies in place that will help guide your decisions.

Conclusion

In this chapter, we started with the navigation of the Power BI service and discussed some of the features for management inside that navigation. We then discussed how to take those datasets and reports we publish and put them in front of other users so that they can share the insights we are gathering.

Next, we're going to discuss some best practices around workspace and app management. We'll also dive into the details around Power BI licensing and when it makes sense to consider Pro versus Premium Per User versus Premium Per Capacity.

CHAPTER 9

Licensing and Deployment Tips

Congratulations! Your report was a huge hit. You took disparate data from different sources, brought that data together, and provided insights from that data that pushed your organization forward. You are keeping your PBIX file up to date, adding new reporting features and measures, and maybe even correcting the odd error or two.

Then you finally experience what every business intelligence professional both hopes for and inevitably dreads. You unlock the imagination of the organization. Now they want you to spread the Power BI gospel across the organization so that other parts of the org can not only leverage what you've built, but also build things for themselves.

First, take a deep breath. Next, let's talk through key questions that will help you put the pieces in place to empower your organization on its larger Power BI journey. What kind of licensing structure should we pursue? What's Pro versus Premium? When does Premium capacity make sense? How do we manage our workspaces and our apps? How do we assign users to roles we created for RLS? How do we make sure our database administrators (DBAs) don't kill us by having a crazy dataset refresh schedule!? Let's talk about it.

Licensing

A *license* itself is a simple idea. It's a purchased agreement that allows you to use a service for as long as you continue to pay for it. Some products have different levels of functionality inside that service, and you may choose to pay for one level of functionality or another. Power BI licensing can be broken into three major categories for most users: Pro, Premium Per User, and Premium Per Capacity.

Pro Licensing

Remember what you get for free. Power BI Desktop is free. Your own private workspace in the Power BI service is free. Your limitation is that you just can't share anything from that workspace. The inability to share can leave some smaller enterprises in a bit of a pinch.

Every user who wants to be able to share their data products in the Power BI service will need a Pro license. In addition, every user who wants to use the data products that have been shared with them will need a Pro license.

Pro licenses, as of the time of writing, for *normal businesses* (those that are not in a government cloud or nonprofits) cost $9.99 per user per month. So, call it $10 per month, or $120 per year per user.

If, in an ideal world, you'd have 10 users and 1 content creator, you're looking at a little over $1,300 for 12 months of coverage. That cost is not insignificant. Certainly many enterprises are looking for ways to cut costs, but Power BI's pricing model, especially for Pro licensing, is very competitive compared to its major competitors like Tableau, Qlik, or Domo.

That doesn't change the fact that it can be tempting to just keep using Excel. Remember that Power BI is more than just an individual component; it's a platform of products built on top of the most powerful analytics engine in the world. Don't undervalue that in your workflow!

If you're in that situation, I recommend doing a real check of who needs to have access to Power BI reporting and, if needed, separate those reporting needs into groups of people who might use department-wide access, if that's feasible based on security needs.

Power BI Pro for *nonprofits* is currently $3 per user per month, a price designed to encourage utilization in that space and help Microsoft look good, for sure.

Power BI allows *individuals* to purchase Power BI Pro licensing. Plus, Power BI gives every user a 60-day free trial at the Premium Per User licensing level, which has all the Pro features. Leveraging that deal might be a way to test who in the organization would most often leverage the Power BI service.

You should be aware of some key restrictions in Power BI Pro licensing:

- Power BI file size limit of 1 GB.
- Eight maximum refreshes for a dataset per day.
- No integration to Azure AI tools.
- No access to deployment pipelines.
- No XML for Analysis (XMLA) endpoint read/write access.

- Total workspace storage cannot exceed 10 GB across all workspaces and cannot exceed the number of Pro licenses times 10 GB.
- Cannot deploy paginated reports.

Premium Per User Licensing

Let's say, for sake of argument, that Pro licensing wouldn't be a problem. In what cases does a Premium Per User licensing scheme make sense? Premium Per User is designed as the middle ground between the Pro licensing and Premium Per Capacity levels. Premium Per User licensing is twice as expensive as Pro licensing.

So, when does Premium Per User make sense? First, do you have a need for any of the items listed in the preceding bullets (that show which features of the Power BI service are limited in a Pro license)? If you do, that's probably a case where Premium Per User makes sense. You can fit a lot of data into a 1 GB size PBIX file.

You may have use cases where that's not enough in a single model. There is a limit of 10 GB for an entire workspace, but most of the time users don't have 10 separate 1 GB files. They usually have a single massive dataset that might be more than 10 GB itself that causes this to be an issue, and this large model support is supported in Premium Per User workspaces.

If you are in a case where DirectQuery can't cut it and you need up to 48 refreshes per day, then Premium Per User is an option there as well.

If you are working in an environment where you have machine learning models in Azure Machine Learning studio or have models running in Azure Cognitive Services and you want to put those results directly into your data model from those services, then, at the least, you'll need a Premium Per User license.

Do you want to use deployment pipelines to manage your PBIX models in a way that's more consistent with continuous integration and continuous deployment (CI/CD) principles? Do you want places to work in development before going to testing and finally production? In this scenario, you would need at least a Premium Per User license.

Do you want outside software to be able to connect to your Power BI data model that you have in service? Do you want to be able to edit a dataset in service without having to mess with a PBIX file? In those cases, you would need XMLA endpoint access, and that can be done with read access or both read and write access. You can't do read and write unless you are in at least a Premium Per User workspace.

If you have paginated reports currently in SQL Server Reporting Services, you can host those reports in the Power BI service. Hosting RDL files requires at least a Premium Per User license.

Here's the catch. If you have a Pro license, you cannot access a Premium Per User workspace. You can't even get around this, for instance, by using an app, because if a given report uses a dataset that is housed in a Premium Per User workspace, that specific report won't work. Microsoft saw that loophole before the release went out. Microsoft does let you add users with Pro licenses to subscriptions, however, so if you have email subscriptions that send a PDF of the report pages, these can be shared with Premium Per User and Pro licensees alike, though they will be static.

Again, in any scenario where you're using Power BI Pro and Premium Per User licenses together in your environment, make sure you ask the right questions to get the users the appropriate licensing based on their needs and to minimize your spending.

Premium Per Capacity, the Big Boy

The final licensing structure I'm going to discuss in this chapter is Power BI Premium dedicated capacity. Premium Per Capacity consists of all the Premium Per User features along with its dedicated compute and dedicated memory. What does that mean? It means the processing cores, the memory, and hard drive space used in the Azure data center are set off to the side and can't be used by anyone but your organization.

Technically, the Power BI service is a generally shared-compute service. That means that Power BI has however many thousands of processors and thousands of GB of RAM that exist in a compute cluster. Therefore, depending on demand in that compute cluster, things can go faster or slower. By pushing your Power BI instance to Premium Per Capacity, you are putting aside Azure resources that cannot be touched by anyone else.

The analogy I like to use when describing this is the difference between a cable modem and fiber optics directly to the home. In a cable modem, your connection goes to a central hub, and all the people who connect to that hub technically share the internet pipe (however large it is) to that hub. When you have fiber directly to your house, it's yours and it belongs to only you. You're not sharing that with anyone else.

Premium Per Capacity comes in different tiers of investment. The licenses for Premium Per Capacity go from P1 to P5. P1 starts at about $5,000 per month. Premium Per Capacity allows you to share Power BI content with anyone, including people not even in your organization, without the need to purchase a user-based license. I would note that your content creators still will need their own separate Pro license.

In addition, Power BI Premium doesn't exclude you from utilizing Power BI's shared capacity with the normal licensing structure. In other words, you can have both Premium Per Capacity workspaces (with all the functionality Premium Per Capacity brings) alongside what I call "regular" workspaces (which users can access with user-based licensing) in your organization's Power BI tenant.

Premium Per Capacity is really designed for enterprise business intelligence at scale. Everything that you can do in a Premium Per User workspace can be done with a Premium Per Capacity workspace. It is also one of the two ways to license Power BI Report Server, which is the analogous on-premises version of the Power BI service.

If you are part of an organization that, for some reason, simply cannot do a cloud-based solution, you can still leverage Power BI and sharing with the assistance of the on-premises product. The other way to license Power BI Report Server is to license SQL Server Enterprise Edition with Software Assurance.

Beyond the Premium Per User workspace, Premium Per Capacity has the following additional features available:

- Ability to share Power BI content with anyone, including people outside your organization if allowed
- Access to Power BI Report Server on premises
- Up to 48 refreshes per day for a given dataset
- 400 GB model size limit (which is *a ton* of data)
- Multigeography deployment for international companies
- Bring your own key (BYOK) functionality
- Autoscale capacity availability
- 100 TB maximum capacity storage

We don't need to go through every feature on this list. For example, if you know you need BYOK functionality, then you already know. But if you don't know what BYOK functionality is, you probably don't need to worry about it.

Is Premium Per Capacity right for you?

The ability to share content with, in theory, an unlimited number of users gives you the opportunity to consider whether Premium Per Capacity makes economic sense. The following is my idea of the formula a nongovernment for-profit organization can use in making such a decision:

> Power BI Cost/Month = (Number of Pro License Users × 9.99) + (Number of Premium Per User License Users × 19.99) + Opportunity Cost of Premium Per User Features for Non-Pro users + Opportunity Cost of Premium Per Capacity Features

You'll notice that I hit on opportunity cost twice. The *opportunity cost* is the invisible cost associated with not having access to those features. For instance, what does it cost your organization to *not* be able to manage Power BI datasets with XMLA endpoint connectivity? What does it cost your organization to *not* have multigeographic redundancy management? These are questions I can't answer for you. But your

organization should think fully through those opportunity costs before deciding if Power BI Premium is the right choice. You can't do a proper return on investment calculation without thinking through the opportunity costs.

It's not simply about the bill you pay Microsoft. It's about the bill you pay Microsoft, plus the costs of having to manage these opportunity costs.

Small business licensing example

Let's work through this formula for both a small business and a 10,000-employee multinational company.

For the small business, let's say you have 10 users. One is a data scientist who wants to be able to integrate some insights from their Azure Machine Learning model directly into Power BI, and those results need to be seen by 2 of those 10 people.

This means we need seven Pro licenses and three Premium Per User licenses. So, we go through the math here and get something that looks like this. I'm going to round the numbers, for ease of math:

> (7 × 10) + (3 × 20) + Opportunity Cost for Pro Users Not Having Premium Per User Features + Opportunity Cost for Not Having Premium Per Capacity Features

In this case, let's say that two of those users should also probably have Premium Per User licenses, and the opportunity cost for those users is twice the Premium Per User licensing fee. And our organization isn't large enough to really worry about most of the Premium Per Capacity features, but some governance things would be nice if they were universal, so let's say that is $100 a month.

Now our formula is (7 × 10) + (3 × 20) + (2 × 40) + 100 = a bill of $130 from Microsoft for the month and a full cost, including opportunity costs, of $310 per month. You can see that we're nowhere near the point where getting Premium Per Capacity licensing makes sense.

Multinational organization licensing example

Now let's look at our multinational company. Out of 10,000 employees, we believe about 3% would use Power BI in some capacity. Out of those 300 employees, 10% would use Premium Per User capabilities. That means so far, our formula looks something like the following:

> (300 × 10) + (30 × 20) + Opportunity Cost for Pro Users Not Having Premium Per User Features + Opportunity Cost for Not Having Premium Per Capacity Features

Before we consider opportunity costs, we're at $3,600 for a bill. If our opportunity costs are greater than $1,400, Premium Per Capacity makes sense. We've already identified we're an international company, and assuming some of our users aren't in the same locations, we really should consider multigeographic redundancy and

deployment management. Is that worth $1,400 by itself? Probably, in that case. With 300 users, should we have some sort of CI/CD-based deployment strategy to prevent data from being siloed? Maybe leverage deployment pipelines? What if we didn't need the Premium Per Capacity stuff, but just the Premium Per User stuff? Well, 300 users at Premium Per User is $6,000 per month, which is above the $5,000 cost of a P1 capacity node.

Obviously, this is a simplified model framework to help you think about whether Premium Per Capacity makes sense. The difficulty is admittedly in figuring out those organizational opportunity costs. However, if you can think through your use cases as an organization, you can usually come up with a reasonable estimation that will help you fill in the blanks in your organization's calculus.

As a quick overview, Table 9-1 neatly puts together the general features that are available at each licensing tier.

Table 9-1. A comparison of the different levels of Power BI licensing features

Feature	Power BI Pro	Power BI Premium Per User	Power BI Premium Per Capacity
Mobile app access	Y	Y	Y
Paginated reports	N	Y	Y
Unlimited sharing	N	N	Y
Report server license inclusion	N	N	Y
Model size limit	1 GB	100 GB	400 GB
Model refresh limit	8/day	48/day	48/day
Azure ML integration	N	Y	Y
XMLA read/write	N	Optional	Optional
Deployment pipeline	N	Y	Y
Multigeo deployment	N	N	Y
BYOK	N	N	Y
Autoscale functionality	N	N	Available as add-on
Maximum storage	10 GB/user	100 TB	100 TB

Workspace and App Management

Regardless of which licensing structure you decide is best for your organization, you'll need to add users to workspaces and to apps that you create, as well. I will be demonstrating all the functionality here with a Premium Per User license.

Workspace Generation and Access Control

In Chapter 8, we briefly showed the interface for workspaces, but let's go into a little more detail here. There are a couple of ways to access a workspace. From the Home menu screen, Workspaces show up as a type of object that can appear in your Recent list, as shown in Figure 9-1. Also, on the menu on the left, a line divides Learn from the Workspace Management navigation.

If I click the menu arrow next to Workspaces, a second gray vertical box will appear, showing the list of workspaces I have access to, and a "Create a workspace" button will contextually appear, if I have the ability to create one, that is.

The most common scenario where you wouldn't be able to create a workspace would be if you were on Premium Per Capacity and the tenant administrator disabled this functionality.

Second, take a look at the vertical menu arrow on the line beneath that Workspaces line. It shows you the workspace you are currently in, and when you click it, it details all the dashboards, reports, workbooks, and datasets in that workspace. We can see what this looks like in my personal workspace example, shown in Figure 9-1.

Here, you can see that I'm currently in "My workspace" and have a couple of datasets. Along with the details of what's in the workspace shown, there is a scroll bar because you can't see all the objects in the workspace from here.

Another way to see all the elements inside a workspace is to select a workspace on the right side of the menu under Workspaces, as shown in Figure 9-1. That will take you to a workspace landing page that shows all the various data elements stored in that workspace.

Now, we can get into workspace generation. You'll note in my previous example that I don't have a Cool School University workspace. I've been working in my personal workspace on the dataset, as you can see it in the list of objects in Figure 9-1.

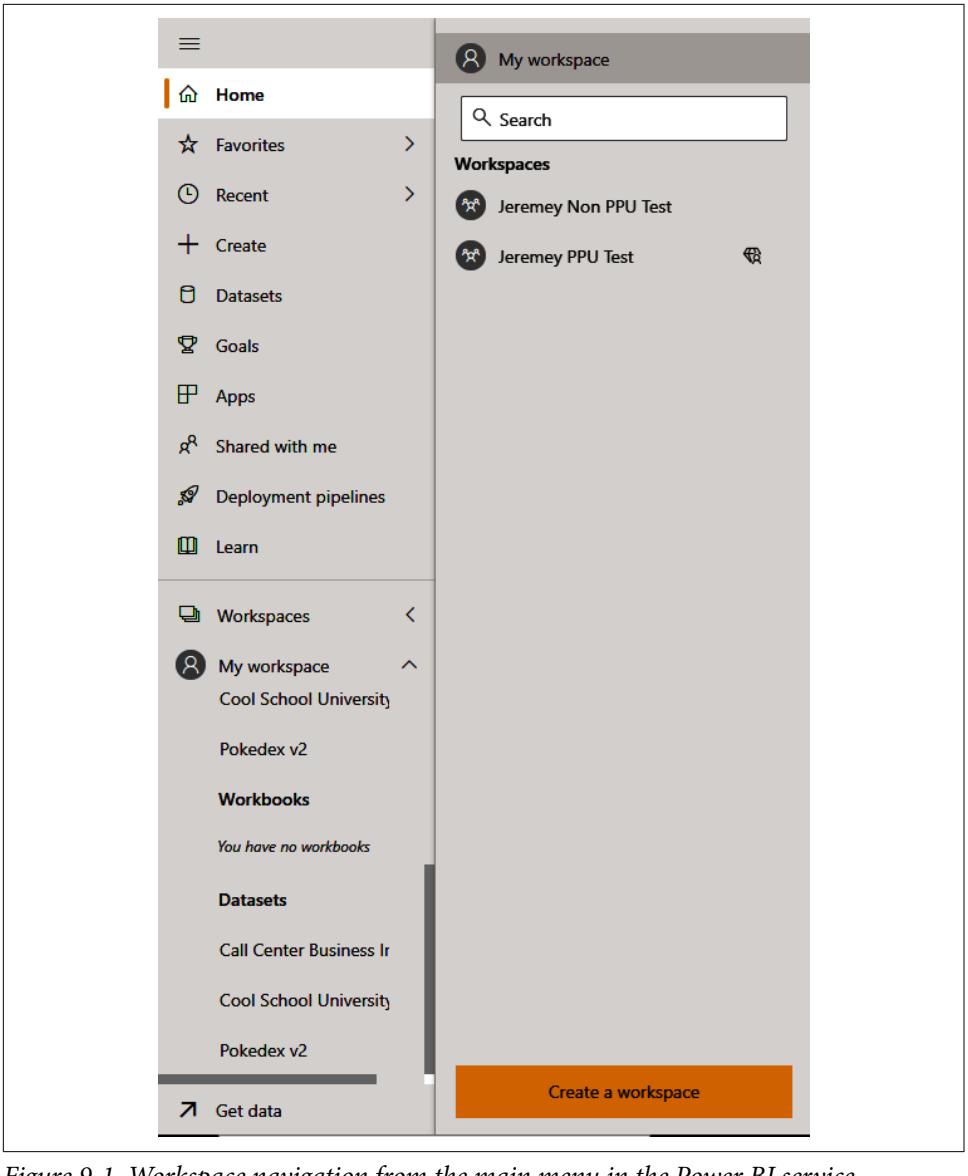


Figure 9-1. Workspace navigation from the main menu in the Power BI service

So, let's create a new workspace using that yellow "Create a workspace" button. When you click that button, a new pane opens on the right side of the screen, detailing the workspace creation options. By default, the advanced options are not shown, so you will need to click the Advanced menu arrow to show these options. We can see this in Figure 9-2.

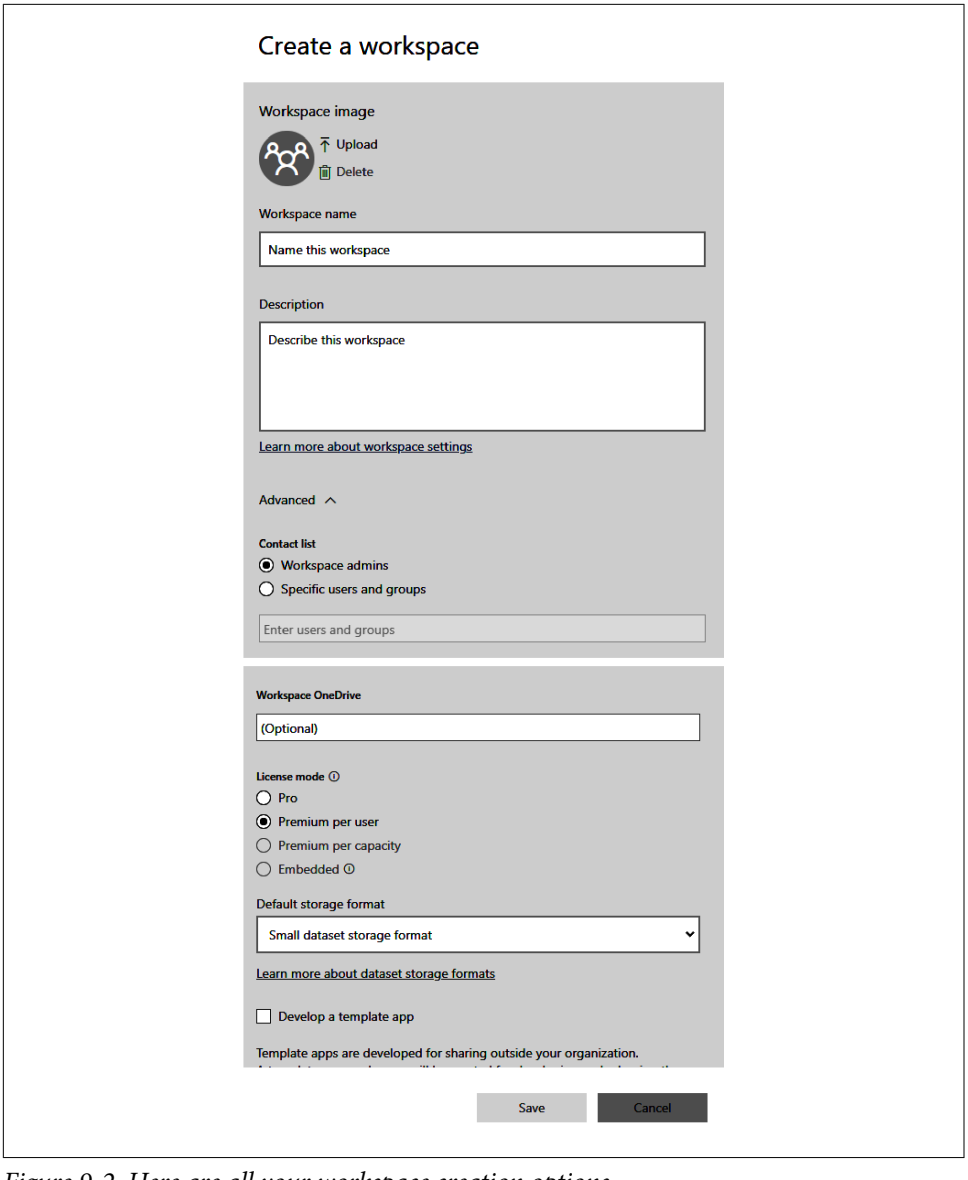


Figure 9-2. Here are all your workspace creation options

In looking at Figure 9-2, going from top to bottom, we can see there's an option to upload an image for the workspace. You can provide a description of the workspace for users.

Then we get to the Advanced settings. First, you choose who is to be contacted with requests for access to a workspace. You can also define a Workspace OneDrive

location for file storage, the license mode of the workspace, if the workspace will generate a template app, and finally if contributors can update the app generated from the workspace, if one exists.. Most of the time, workspace administrators will be the people you want notified for access requests. However, in Premium Per Capacity environments, the tenant administrator can exercise greater control over workspaces, and the organization may have all such requests routed to an IT group or tenant administrator in such circumstances.

We can set a OneDrive for Business location, to be used by the workspace as a place to store documents. You can't, however, just use your personal OneDrive. That would be too convenient for private users, wouldn't it, Microsoft? So what you really target here is the Microsoft 365 group's SharePoint document library.

This group should be created outside Power BI first. You can, in fact, create a group like this from OneDrive for Business, but the ability to create Microsoft 365 groups can be restricted in your environment. If it is restricted, reach out to your IT department or SharePoint administrator for additional details on linking OneDrive and your new workspace.

Next, we have the "License mode" selection. For our purposes, this will be the most important selection, since it determines the functionality our workspace will have. In the previous example, I have two options available to me: Pro and Premium Per User. I don't have Premium Per Capacity because that's outside my personal budget.

If you choose Premium Per User or higher, one additional element appears that was not previously there at all, in what can only be considered inconsistent design language. You are asked to choose a default storage format for your datasets. You can choose either the small dataset storage format, which is the default for all Pro workspaces and can't be changed, or the large dataset storage format. If you plan to have data models in excess of 1 GB, make sure you have "Large dataset storage" enabled.

Next, you can choose whether this workspace will be home to a template app. A template app is developed for sharing content outside your organization. If you have a need to develop a template app, I highly recommend beginning with Microsoft's documentation for template apps at "What Are Power BI Template Apps?".

The final choice is also important if you plan to use elements from this workspace in an app eventually. We will discuss access levels and controls after this, but this question is asking whether someone in a contributor role should be able to edit an app that is powered from this workspace. That is a data governance question for the workspace administrator and/or tenant administrator to answer. In some cases, like in development, you may want to enable that function so that you can make changes quickly. But you may want to turn it off when it goes into everyday production to minimize the number of people who could disrupt operations with accidental edits.

Once everything is filled out, that Save button will turn yellow and the workspace is created. You'll note that Pro workspaces do not have a diamond next to them. Premium Per User and Premium Per Capacity workspaces will always have that diamond next to their names to help users identify the kind of workspace it is.

Managing Users in a Workspace

I've created a Premium Per User workspace titled Cool School University, and now I need to add users to it. From the landing page of any workspace, toward the upper-right corner of the page, are options for View, Filters, Settings, Access, and Search. Access is what we're looking for. When we click that button, a pane similar in form to the Workspace Generation pane slides in from the right, and that's where we can add users. We can see this pane in Figure 9-3.

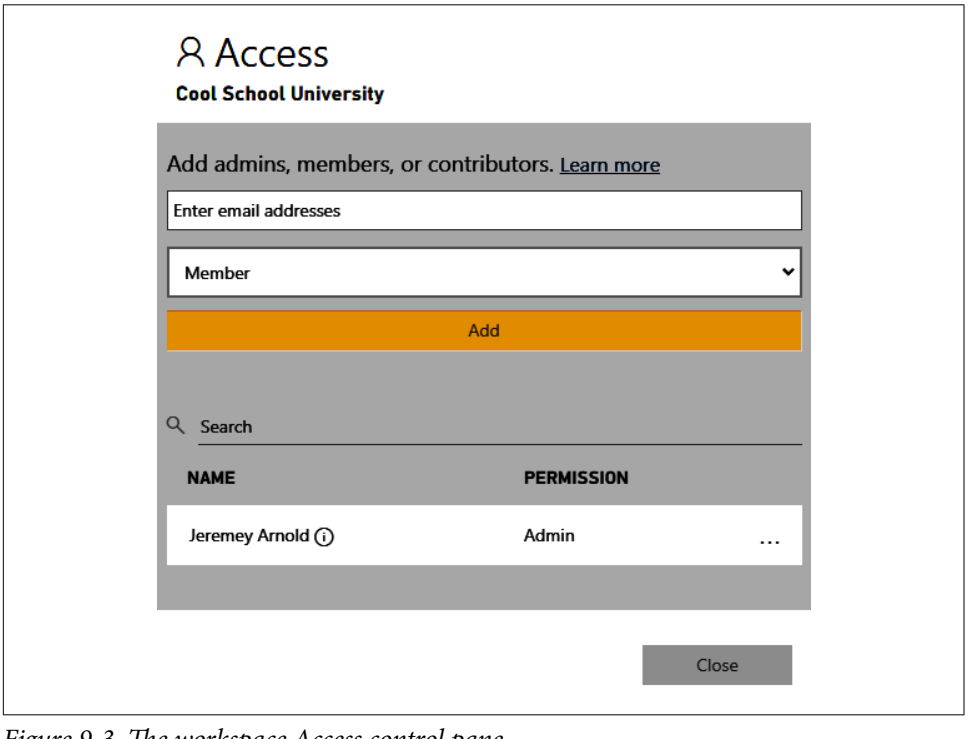


Figure 9-3. The workspace Access control pane

Power BI works very neatly with an organization's active directory instance such that if your organization is using Windows or Azure Active Directory, Power BI can and will search for users as you type their email addresses or Microsoft 365 groups if you have them, as shown in Figure 9-4. Yes, you can add entire groups into a workspace with a couple of clicks.

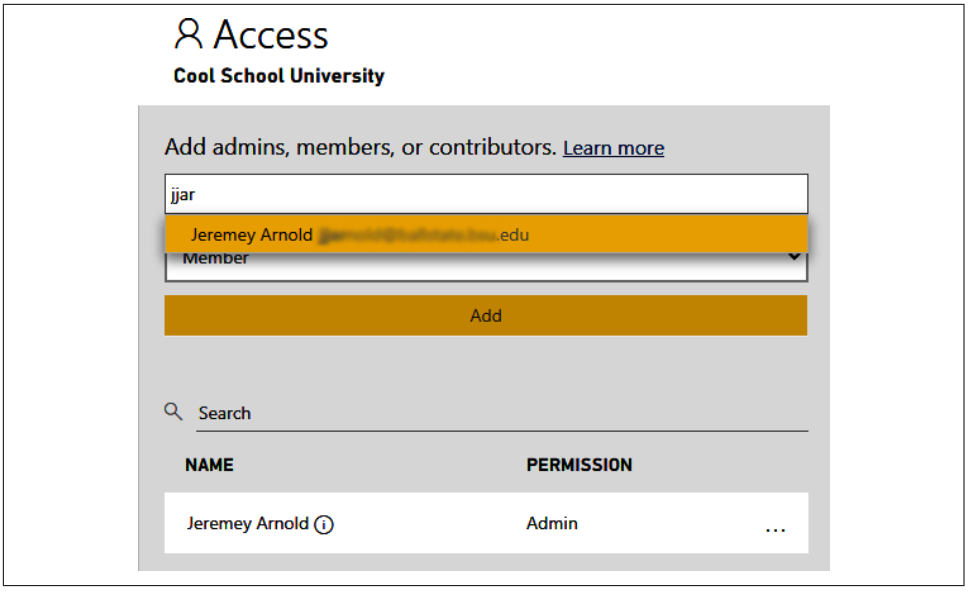


Figure 9-4. Active Directory puts in the work for you by searching for and filling in names or groups as you begin to type them

You can also add multiple people at once by typing multiple emails. Notice that line beneath the email address? That's where we determine the level of access to the workspace a user will have. Four roles can be assigned to users in a workspace: Admin, Member, Contributor, or Viewer. These roles are hierarchal in nature, so someone in a lower role can't affect the user access of someone in a higher role. For instance, a Member can't kick out or remove an Admin from a workspace, but the Admin can kick out the Member. Only Admins can remove other Admins.

Admins and Members can do things that either impact users in a workspace or modify workspace elements that are used outside the workspace. Contributors can work on things inside of the workspace but generally don't have the ability to interact with the way external viewers interact with workspace data elements. Viewers can only see objects in a workspace and cannot make any edits to any objects in the workspace.

My advice when managing workspaces is to have at least one service account, an account that does not belong to a user, but to the organization at large and for whom the credentials are owned by a limited number of people. This service account should be included as an Admin in all workspaces, even if it's not generally used. If for some reason the only Admin of a workspace left the organization, you would have to go through the Power BI Admin API at that point to promote a new Admin to the workspace. I think that's a bit annoying.

However, your organization may have data governance protocols that will help you determine the best course of action for you and your organization in this regard. The most important thing is to have some way or method in place so that, if necessary, you can regain control of a workspace in the event of a personnel disruption.

With that wonderful segue, let's quickly discuss how you remove a user or change their role in a workspace. In Figure 9-5, to the right of the listed permission, you see an ellipsis, three dots. Click that, as a button, and a small pop-up appears that will show a list of roles you can reassign to a user. At the bottom of that list, you'll see Remove. If you click Remove, they're gone. If you click another role, that user is moved to that role. That part is easy, maybe a little too easy, given that Power BI does not even bring up a confirmation message asking you if you're sure you want to do this. However, if you make a mistake, you can easily fix it, as you can see from how easy as it is to add users.

Adding Users to Roles for RLS Implementation

You have your workspace set up with the users who should have access. Now you need to add them to RLS for each dataset in the workspace that has different roles defined. This is pretty simple to do, as the hard work is really in defining the roles in the first place, as we discussed earlier in this book.

From a workspace's landing page, hover over the dataset you want to add users or groups to for RLS until you see a vertical three-dot ellipsis menu appear next to the dataset name.

Click the ellipses and go to Security. When you do that, if you have roles defined for your Power BI dataset, you'll see the list of roles and the current members of those roles in the dataset. You can add users or Microsoft 365 groups.

One thing to remember, if a user belongs to more than one group, Power BI will treat them as though they have full access to whatever combination of data would be present in between the multitude of roles, much like a SQL outer join.

Figure 9-5 shows the simple RLS user addition interface, and just as with adding users for the workspace, Power BI will attempt to find users as you type in the appropriate email or group name.

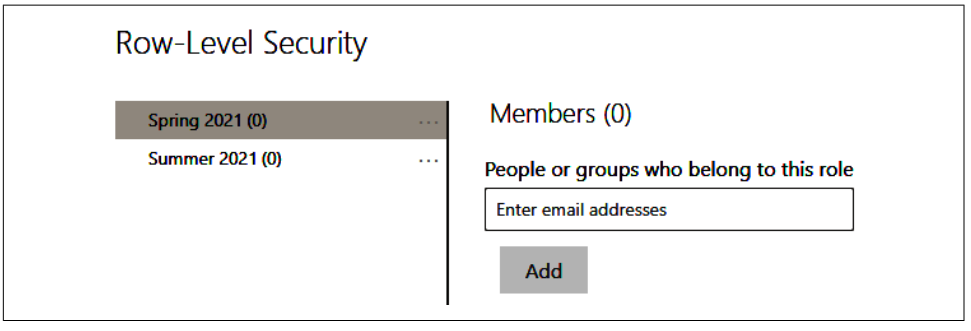


Figure 9-5. Get those users in the roles they belong by assigning RLS

App Creation and Management

At some point, you may want a way for users to interact with your data elements that you've created without adding them to the workspace directly. You can do this by adding them to an app that is powered by a workspace.

From our workspace landing page, a big, yellow "Create app" button is in the upper-right corner. Let's start there and see what we get in Figure 9-6.

Here, in this first Setup page, we have several options. We name our app and provide a description, and we can provide a link to a support or documentation website where users can get help with using the app, provide a logo for our app, pick the app's theme color, and finally we can show contact information for who should be contacted about access to the app. This should all feel familiar at this point, as this is in line with our workspace creation experience.

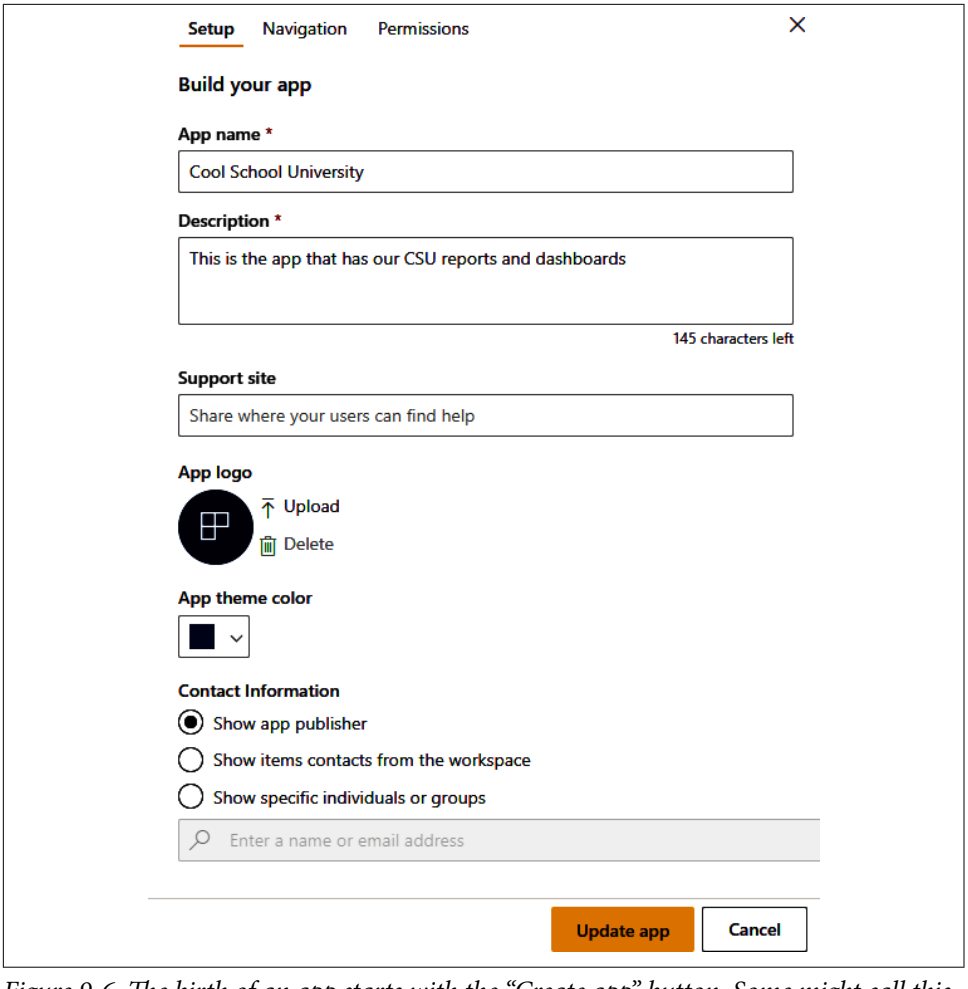


Figure 9-6. The birth of an app starts with the "Create app" button. Some might call this app setup.

However, you'll see some other items we should manage before clicking that "Publish app" button. Next to Setup, we have Navigation and Permissions. Let's tackle those in order, starting with Navigation, as seen in Figure 9-7.

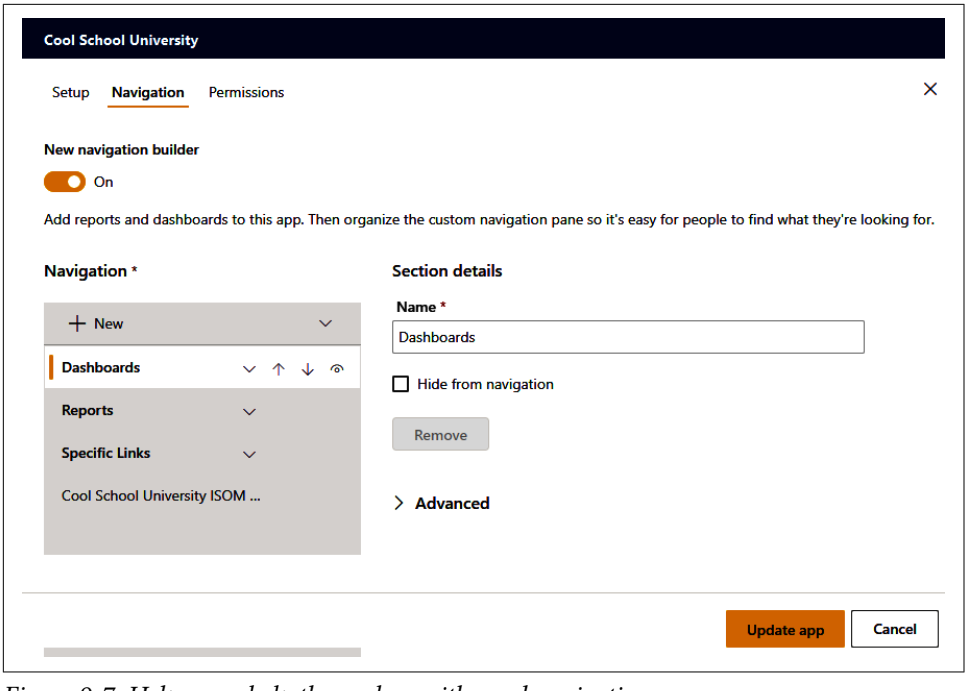


Figure 9-7. Help users help themselves with good navigation

The first thing we see is a on/off toggle to let us choose whether we want to use the New navigation builder. At some point, the old option, I imagine, will just go away; if you are starting new with apps, provide your users a consistent experience and just leave this setting alone.

In an app, we do not include any of the underlying core data, only the data products. We don't include datasets or dataflows in an app, for example, but we do include reports and dashboards.

You will see on the Navigation pane on the left that all the dashboards and reports that exist in the workspace are put into the app by default. In this case, I have a generic dashboard that was created when I uploaded my PBIX file, and I have the actual report itself. The gray menu on the left that shows what those elements are. You don't really *remove* items from an app so much as you *hide* them. You can hide items that you don't want included by clicking the little eye next to the up and down arrows. If it has a line through the eye, it will be hidden.

Navigation becomes useful when you click that +New button and can either add a section or a link. A *section* is like a display folder. You cannot have a multilevel folder structure—no folders within folders here. A *link* gives you the ability to define a link to a web page, either in Power BI or not, and what the browser behavior should be to open it.

When you create a section, the area on the right in Figure 9-7 will give you places to name the section and give you the opportunity to hide the section from navigation.

When you create a link, you name the app element, provide the URL for the link, determine how that link should be opened when selected, and then choose a group for the element to fit in for navigation.

OK, so at this point, you have your navigation set up all nice and neat. Now you need to figure out who should be able to access the app *and* what underlying permissions are granted by having access to the app and the datasets that power the app behind the scenes. That's a lot to take in, so take a moment.

We can see the Permissions window in Figure 9-8.

As you can see, first you must decide who should have access. Is it the whole organization? Is it specific individuals and/or groups in the organization? In this case, *groups* refer to Microsoft 365 groups. Power BI also notes that anyone who has access to the workspace will have access to the app. That makes sense for the same reason a workspace Admin isn't affected by RLS. A workspace administrator can just go download the PBIX. Anyone who has access to the workspace could just go to the workspace and see the material.

Next, we choose whether people who have access to the app will be able to connect to the underlying dataset and if they'll be able to make copies of the reports in the application in their own workspaces. These two selections are hierarchical. If you don't have access to the app's underlying datasets, you can't copy reports. But having access to the datasets doesn't mean you can copy reports. One is necessary, but not sufficient.

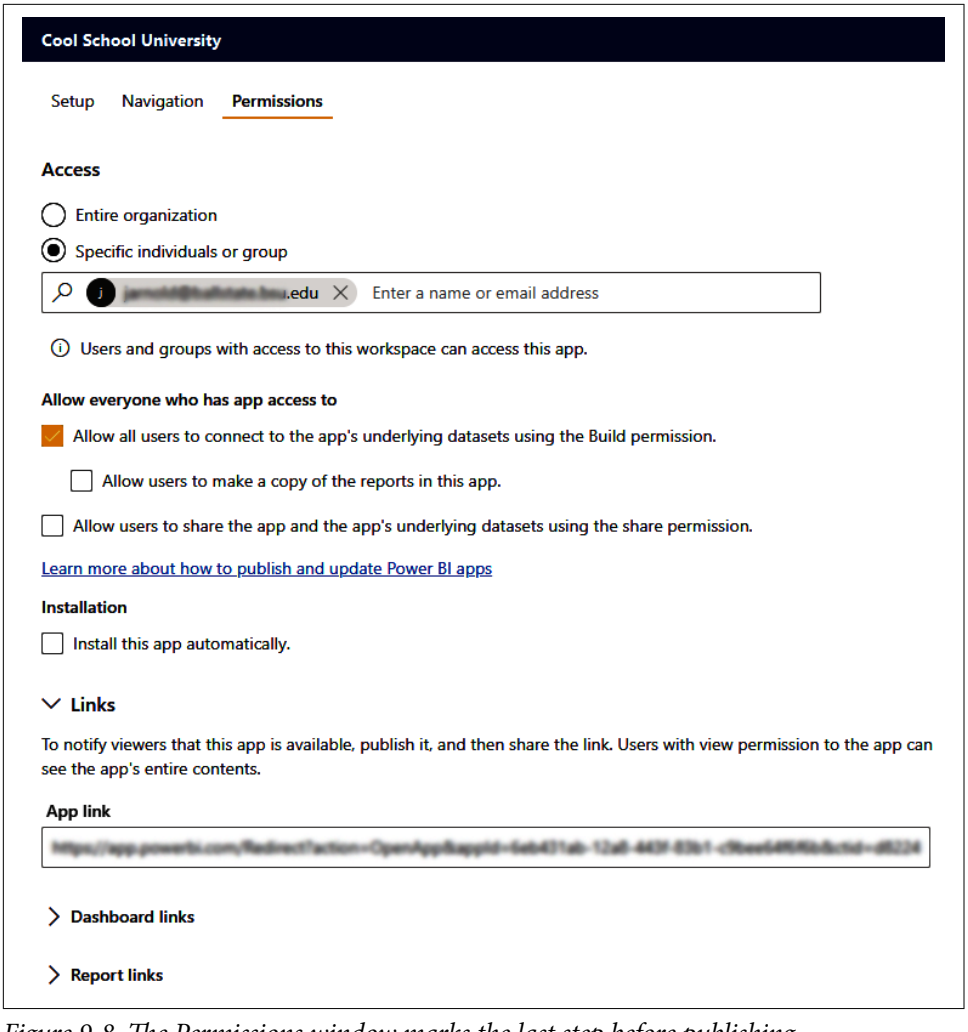


Figure 9-8. The Permissions window marks the last step before publishing

Next, we must choose whether we want users to be able to share the app and the app's underlying datasets with the share permission. It's odd to me that this also isn't hierarchical. Generally speaking, I leave this option off and try to have a location where users can request access to the app from the team that owns the app, as opposed to things being shared with no guardrails.

Finally, there's a link to extra Microsoft documentation and a choice of whether this app installs automatically for users who would have access to it. I don't mind having automatic installation turned on, as that can be really convenient for end users.

After this, we can publish the app! The app will usually take between 5 and 10 minutes to get packaged, and then it will appear in the Apps section in the Power BI Navigation menu. If you need to update the app, come back into the workspace that hosts the app, and you'll see that "Create app" button has changed to "Update app." That's it. You now have an app!

The Golden Dataset(s)

At some point, as you manage your Power BI deployment, you are going to accumulate a lot of data elements. As those data elements increase, you have computational concerns. If you are in a world where every report is powered by its own dataset and each of those datasets is refreshing, that's a lot of moving parts!

Heaven forbid you have that many datasets refreshing against enterprise databases...because if those datasets are of any size at all, you might give your poor DBA an actual heart attack! How much of the data between those datasets is replicated for no reason?

These are questions that, when the Power BI service really came into its own, a lot of analysts who were using Power BI and pushing this space forward didn't have really good answers to. But we have a thousand Excel workbooks floating around, so why is this any different?

The problem is that, at some point, data elements become mission critical. We need to manage them like mission-critical assets. So, to look at how we needed to manage this problem, we needed to go back into Power BI's past and remember what it is under the hood. It's Analysis Services. How would we manage an enterprise-wide Analysis Services deployment? We would create a core master dataset or a small number of datasets that we could manage. Doing that would minimize the computational burden on other parts of our data pipeline.

We can do the same thing with Power BI by creating that master dataset, or golden dataset, or whatever you want to call it. The fewer times we can make big data requests of our databases, the better off we are. The fewer places we must manage RLS, the easier it is to manage. The fewer datasets we must manage, the more we can feel comfortable pushing out our best data elements in front of our data consumers, knowing there's a higher chance of being able to answer the questions users want to ask. We do that by making the data easier to find.

Does this mean you must have one dataset to rule them all? Not necessarily. You could certainly organize data into a data-mart-type structure, creating a small number of datasets that are highly curated to answer specific types of questions but are still reusable data elements.

A good example of this goes back to RLS. In a Power BI dataset, you establish the roles that exist and define those roles in DAX. If you have three similar datasets, that

means you have to manage those roles in three different locations; you have to remember to go into Power BI service, find the dataset in the Workspace landing page, hover over the dataset until the ellipses menu pops up, click that, go into security, and make sure to include the people or groups into those roles you've defined. What happens when you forget to do it once and one of the datasets isn't updated? That means someone sees data they shouldn't.

You could definitely see this being a larger issue in, say, a Premium Per Capacity environment where you might have hundreds or even thousands of report viewers who could be in dozens of groups for which you're trying to manage access.

Does this mean that when your organization gets large enough, you should stop considering Power BI for ad hoc spontaneous analysis? Of course not! Hopefully, you'll be able to leverage your dataset, or small numbers of datasets, to get to the answers you want to find more quickly. However, sometimes you have to go outside the box to look for data.

Microsoft recommends in large Premium Per Capacity environments that customers split their business-mission-critical capacity from their ad hoc exploratory capacity. I think this is a good idea and easy enough to do with workspace management, even in nonpremium environments.

Encourage your users to connect to datasets in the Power BI service as data sources in Power BI Desktop when they start building reports of their own. This will reduce dataset bloat. I have seen too many organizations with dataset bloat decide that Power BI was just too complicated of a tool to use. It's because they didn't have good governance practices in place and eventually got to a point where users lost track of all the resources they had available.

Conclusion

If you've gotten this far, I genuinely hope that this text has helped you feel more confident in your use of Power BI Desktop and, in these recent chapters, the Power BI service. We've discussed licensing, workspace management, app management, and general deployment strategy to help you stay in front of problems.

In our tenth and final chapter, I'd like to expose you to some third-party tools in the Power BI ecosystem. These tools will be helpful when you try to troubleshoot issues and speed up development. Hopefully, they'll give you a leg up on your Power BI development and management journey.

CHAPTER 10
Third-Party Tools

To this point, everything that has been discussed in this book has been relevant to a specific part of the Microsoft Power BI ecosystem, usually Power BI Desktop or the Power BI service. Now we're going to step outside that realm.

As you grow in your Power BI skill set or come across new development requirements, you might eventually find some parts of Power BI that aren't as easy to manage as you'd like. Things like exporting data from Power BI aren't as straightforward as they could or maybe should be. Version control has always involved tedious PBIX files, and while Microsoft may have better internal solutions in the future, it too is currently not as easy as it should be. At some point, you may want a little more granular control over where you put your measures when they're created, or you may want to have them automatically generate in the display folder you've created for them.

The purpose of this chapter is to tell you about some great options that will improve your use of Power BI. Power BI is really a great ecosystem, but no software or service is perfect. Thankfully, independent developers inside the Power BI community have gone above and beyond to create completely free tools that can provide very real quality-of-life benefits in your Power BI journey. There's software to streamline your development experience, manage data models, help with version control, format your DAX properly, and more.

In this chapter, I'm going to highlight some of the most popular external tools used by the Power BI community. I'll give you a brief overview of their functionality and discuss ways you can use them. I don't plan to give exhaustive explanations for every tool. My purpose is just to get you started in learning how to use them in your workflow. You'll be glad you read this final chapter.

Just think, soon you will graduate from Cool School University and make me so proud. But let's keep working for now.

Get to Know Business Ops

For most tools, you can try to find each individual piece or extension, download that extension, install it individually, and manage stuff. I'm lazy and don't mind admitting it. With that in mind, *PowerBI.tips* makes an external tool manager called Business Ops that is completely free to download.

This site does all the hard work of collecting a ton of third-party tools, bringing them into one place, and allowing you to quickly install and configure the ones you are interested in. It's consistently updated when newer versions of the external tools it links to are also updated. This is important in helping you stay current as development improvements are made to those other external tools we'll talk about later in the chapter.

The download itself comes as a ZIP file, with the installer located inside the ZIP file. Once the installation is complete, you'll see an interface like the one shown in Figure 10-1. On the Home landing page, you can see the release notes for the version of Business Ops you've installed. The code is completely open source, so if you want to go to its Git repositories, you can.

Business Ops will allow you to install a ton of Power BI add-ons, easily access learning tools, create Power BI themes, use a gallery for custom visualizations outside of AppSource, and find links to some of the best DAX resources on the planet.

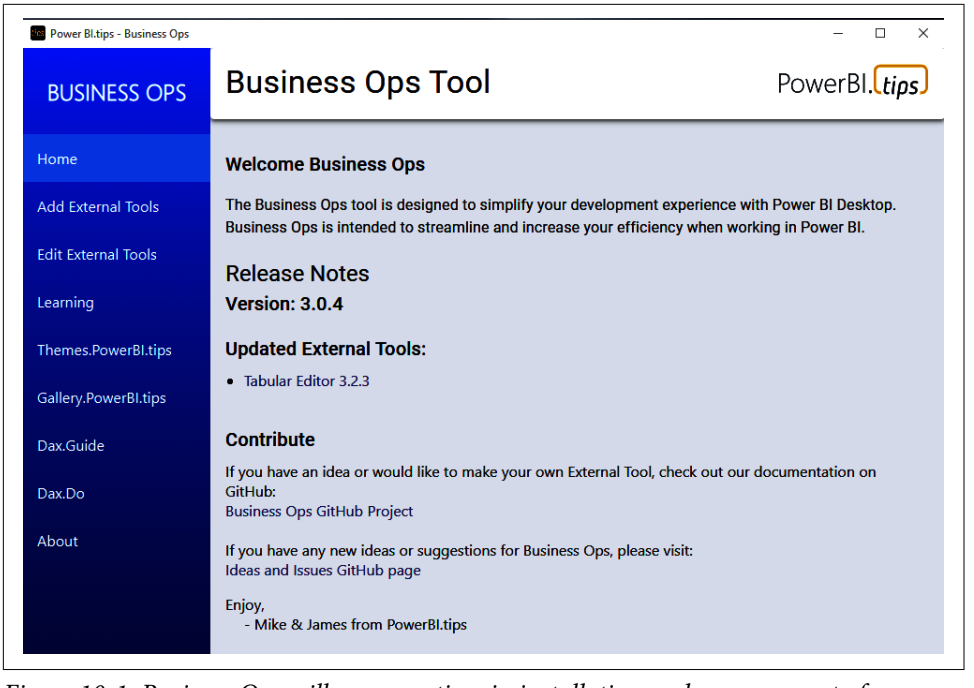


Figure 10-1. Business Ops will save you time in installation and management of your Power BI external tools

Add External Tools, Remove External Tools, and Modify Display Order

From here, we can go to Add External Tools to see the list of external tools that Business Ops has curated, then get those packages added so that Power BI recognizes them as external tools. You do this via a very simple checkbox interface, as shown in Figure 10-2. This page is quite long, and while the scroll bar on the right side of the application isn't the easiest to see, I promise you it's there. I personally like to navigate this with my mouse scroll wheel.

This chapter focuses on the ALM Toolkit, Tabular Editor, DAX Studio, and Bravo. When you have all the tools you want to add, click the blue "Add External Tools" button in the bottom-right corner and let Business Ops take care of the rest.

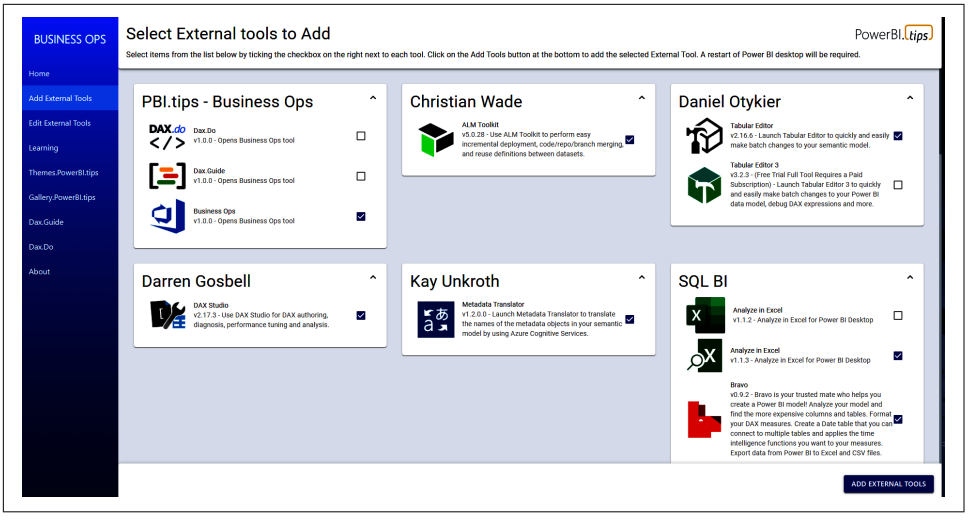


Figure 10-2. Tool installation is so easy with Business Ops

One inconvenient aspect is that the software add-ons, like the ones we are discussing, get installed into the same folder where the Business Ops software is installed. Business Ops also won't make separate entries in your Start menu for these software installations, which is different than if you installed them manually from their individual installers.

I get around this by opening a blank Power BI file after the external tools are added and using those external tools to force open those pieces of software. Then, in the Windows taskbar, right-click and select Pin to Taskbar.

If you want to create individual shortcuts for your desktop, go into Windows Explorer, navigate to the software inside the Business Ops directory, and make a shortcut to the software, as you would normally in Windows.

Under the Edit External Tools section, you can modify the relative order in which the tools appear in the External Tools part of the ribbon in Power BI Desktop. You can also remove components from the list altogether.

When you install external tools, a JSON file is generated for each tool. Clicking the pencil icon allows you to modify the filename. The tools in Power BI Desktop are shown in alphabetical order, left to right. By default, each JSON file starts with a three-digit code that sets the initial order.

In Figure 10-3, I've modified five of the items to have five-digit codes and gave them numbers that will put them in front. I give the ones I edited five digits so that I know I've edited them. By giving them five digits, assuming the first three are zeros, they will always appear before the first autogenerated three-digit code.

You can also remove items from your External Tools list if they were installed via this software. Do this by clicking the trash can next to the pencil/Edit button. If you remove something by accident, you can go back to Add External Tools, and everything you already have there will be grayed out. You can just select the tool you want to recover and add it again.

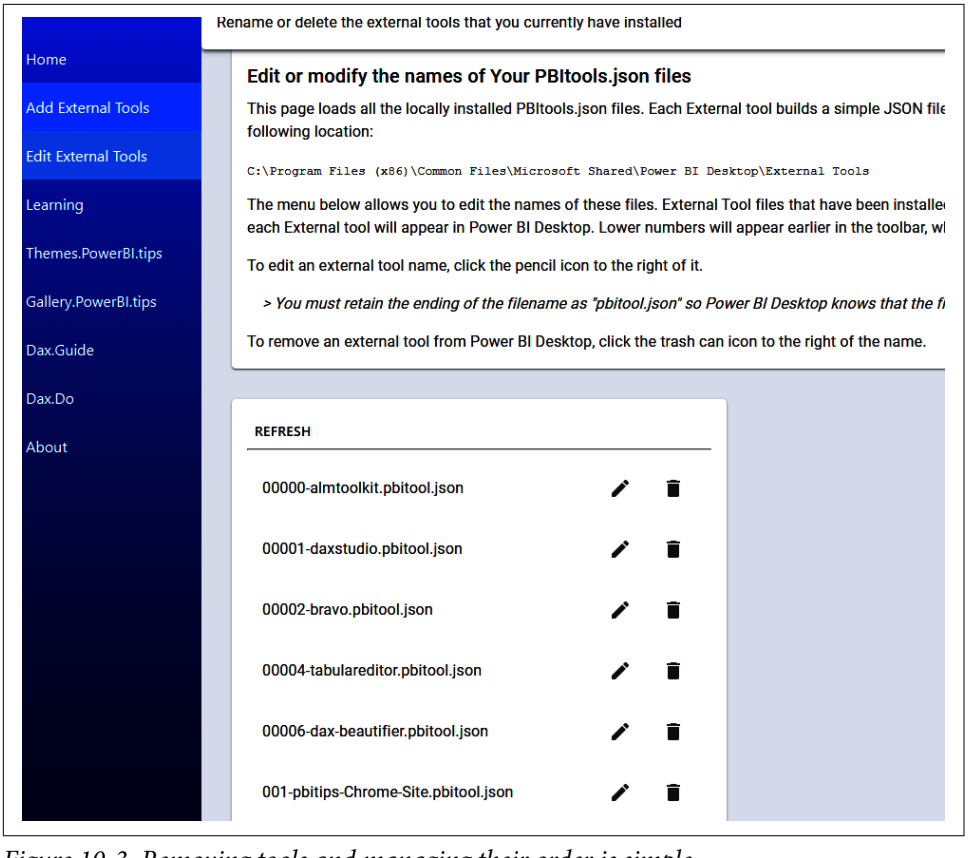


Figure 10-3. Removing tools and managing their order is simple

Learning, Theme Generation, Visual Generation

The Learning section has additional resources about topics we touched on briefly earlier in this book, like "Quick measures" and some things that we touch on later in this chapter and in Appendix B. The Learning section also contains links to documentation and training videos for the additional external tools we'll discuss later in the chapter.

The custom theme generator is incredible! You can put together a theme of colors and custom visual settings that you can set at the global or individual visual level. Then you can import it as a custom theme in Power BI Desktop.

This function will also give you the hex codes for all the colors you choose so that you have those for future reference. Let's first look at the color palette generator in Figure 10-4.

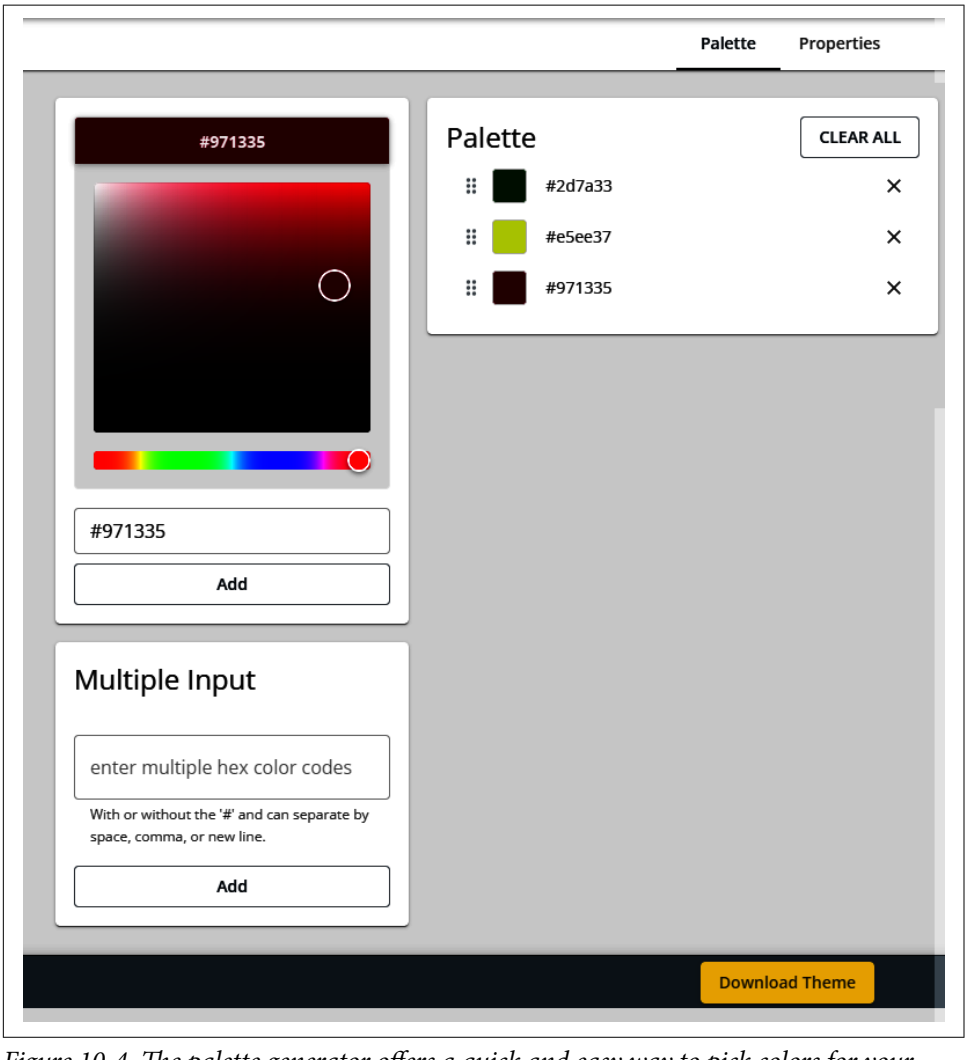


Figure 10-4. The palette generator offers a quick and easy way to pick colors for your reports

On the left, you see a color sliding scale that allows you to go from red all the way to violet and then back again to red in the red, orange, yellow, green, blue, indigo, violet color order. At any point on that bar, the square for the selected color range will show you a variety of lighter and darker shades to pick from, along with the hex code for the currently selected color. If you happen to already know your hex codes, like if

your marketing department gave you your organization's hex-code color scheme, you can enter those codes directly in the input section at the bottom.

In Figure 10-5, we can look to add preselected formatting options on either a global or visual-by-visual basis as a part of our custom theme.

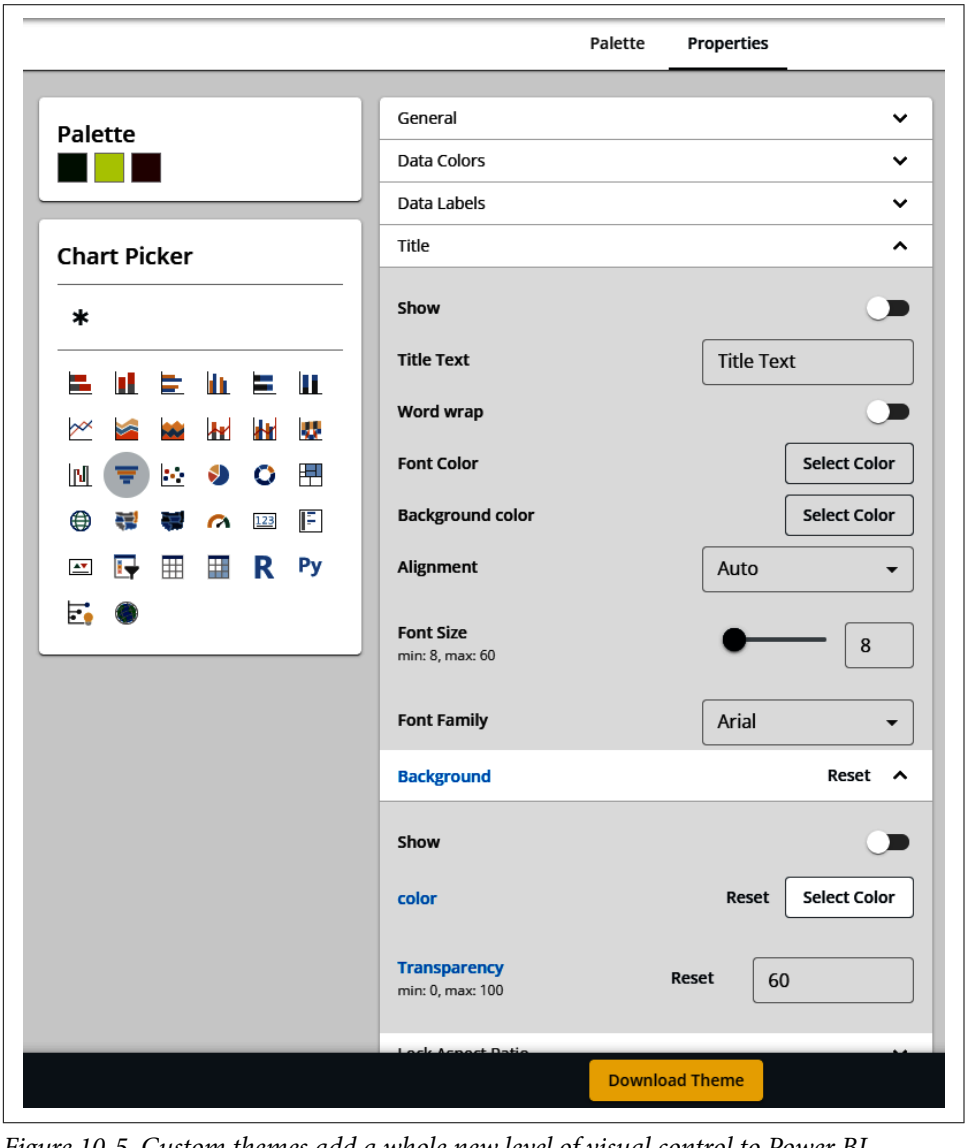


Figure 10-5. Custom themes add a whole new level of visual control to Power BI

Here you can see that the colors I chose previously from Figure 10-4 are still present. In this case, I have the funnel chart selected, and you can see the variety of formatting options I have on the right. For any value you've modified, the text of that category and the changed options in that category will display in blue text.

In this exercise, I've modified the background portion of the formatting options by specifically changing the background color and the transparency level. The * in the Chart Picker will apply changes made there to all the visuals where applicable. When you have your colors and your custom formatting the way you want them, you can download the theme as a JSON file and import the theme into your report in Power BI Desktop. Do this by going to the View portion of the ribbon and, in the Themes drop-down, select "Browse for themes." You'll navigate to where your JSON file is on your computer, select it, and that's it!

It's important to know that if you make changes to a visual's base formatting options at the theme level, they can still be modified as an individual visual on your canvas in Power BI Desktop. This is more about accelerating a change when you know how you want the visual to behave a majority of the time when it's different from the default formatting preset.

In the Gallery section, you can find a list of color themes that have already been generated in the Palette section. If you click a color theme in this portion, it will bring up a small example Power BI Report page that will show you what the theme looks like against visuals. I find this feature incredibly helpful.

The Charts section shows you a list of custom visuals that have been created with Charticulator, a do-it-yourself visual creation tool created by Microsoft. You can download and import these custom visuals into your Power BI Desktop report. Do this by adding it as a custom visual by importing the PBVIZ file. We can see this in Figure 10-6.

The Create My Own option will open a fully functional version of Charticulator inside the application. Note that this is technically a *fork* (or a copy of the code repository) of Microsoft Charticulator, as Microsoft has made some small UI edits. But anything you build in this version of Charticulator will behave in the exact same way as if you had built it by going to the Charticulator website.

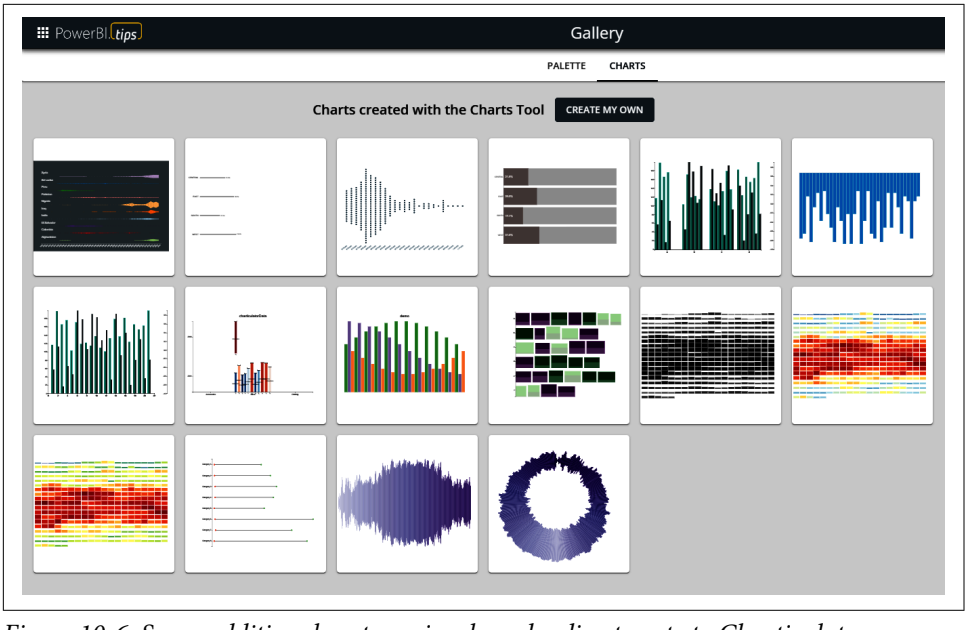


Figure 10-6. Some additional custom visuals and a direct route to Charticulator

Additional DAX Resources

Business Ops links directly to two additional fantastic DAX resources, DAX Guide and DAX.do, both courtesy of the folks at SQLBI.

DAX Guide provides a list of every DAX function and includes its syntax, the kind of values that function returns, real coding examples written with DAX best practices, and, for many functions, a YouTube video in the upper-right corner that has even more details on the function in a friendly video format.

On the home page of DAX Guide, as shown in Figure 10-7, we can select a specific group of functions to look at or learn about the most recent DAX functions that were released and when. As someone who works with DAX in more contexts than just Power BI, the ability on the left to see which products a particular DAX function will work in is a lifesaver!

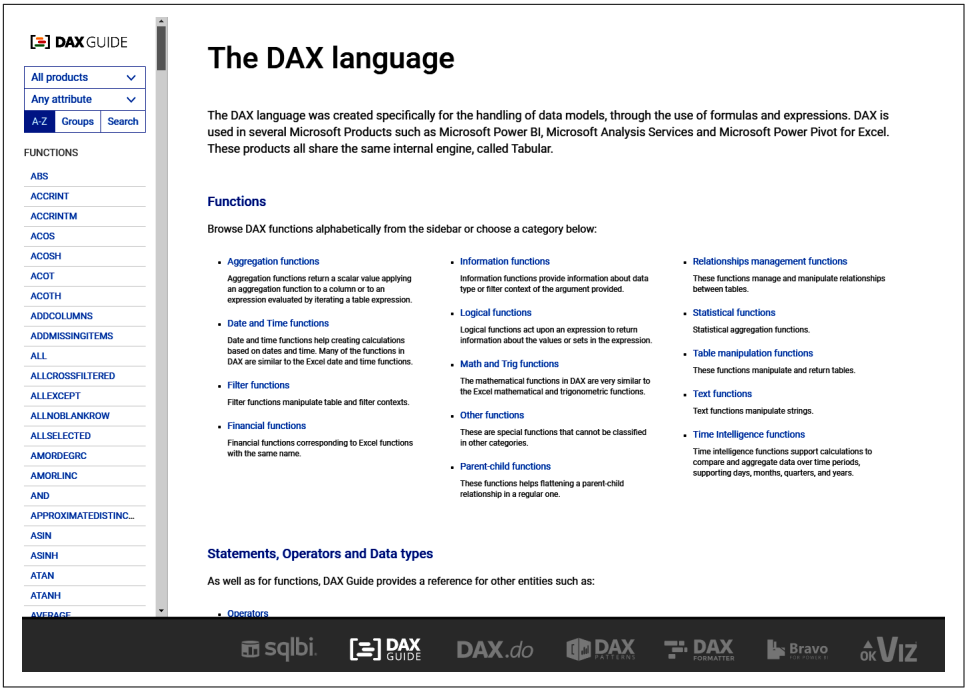


Figure 10-7. DAX Guide is an amazing combination of knowledge accessibility and real-world examples

DAX.do, on the other hand, is a full-featured DAX playground that allows you to manipulate and move around certain elements to fit your preferred testing style. The playground comes with two data models that you can switch between, Contoso and the DAX Guide models. DAX.do has drag-and-drop functionality from the column list. It identifies which functions you are trying to use and will show you a drop-down list of those functions to bring up their DAX Guide pages. You can see your results, and when you write a query with an error, you'll get an appropriate error message.

Figure 10-8 shows DAX.do and its component parts. DAX Guide and DAX.do have their own websites, so if you would rather engage these tools from a browser, you can.

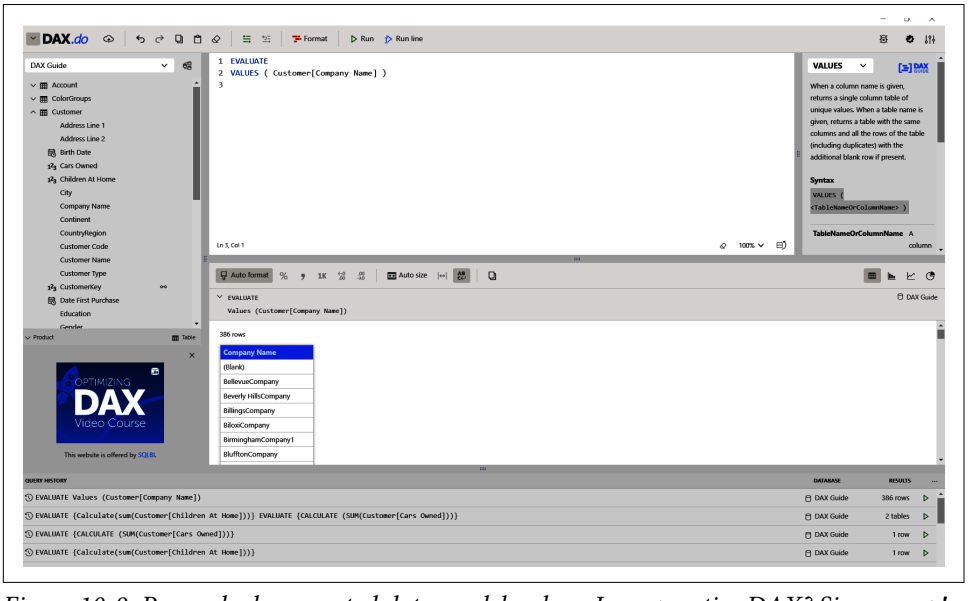


Figure 10-8. Premade documented data models where I can practice DAX? Sign me up!

With Business Ops giving us access to a wide range of third-party tools, I want to highlight some of the most useful ones in the next section. In particular, I'll cover a DAX editor, a dataset editor, and a tool for reviewing dataset health and easy measure generation. First, let's talk about DAX Studio, the preeminent DAX querying tool.

DAX Studio

DAX Studio is a SQL Server Analysis Services–based (SSAS) tabular model query tool. It allows you to write DAX queries against a data model and get results. When it is opened directly from the External Tools portion of Power BI Desktop, it will automatically connect to the Power BI dataset it was launched from. You can also use this to query datasets that are in the Power BI service if you have the XMLA endpoint enabled to do so. Let's look at the interface in Figure 10-9 and discuss what is available.

We won't go over every option here, but there are a couple key things to find. First is the large Query window in the center of the screen. That's where you write DAX and get results back. At the bottom, you can see the results that were returned. On the left, you see my list of tables and columns, along with any display folders I have.

Along the top of the ribbon are options to run a query you write, cancel the query that's being evaluated, clear the in-memory cache, and determine an output format for your results.

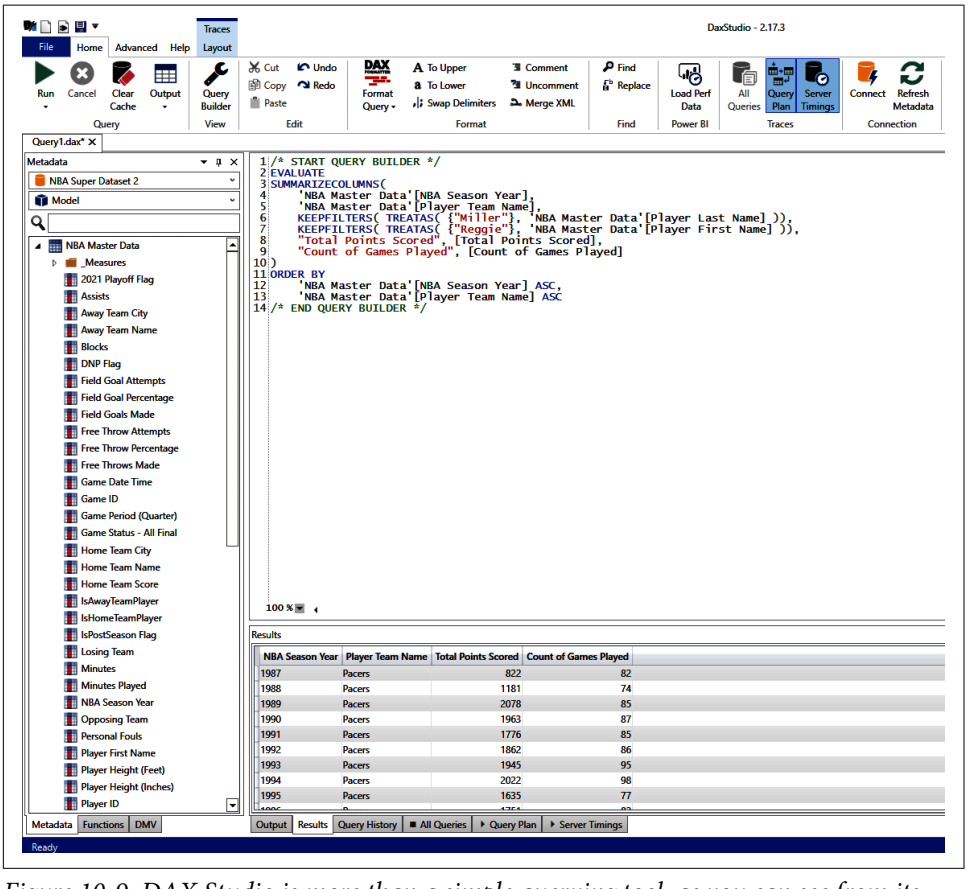


Figure 10-9. DAX Studio is more than a simple querying tool, as you can see from its interface

I'll skip over the Query Builder for a moment, and you can see that classic clipboard functions are made explicit in this ribbon. There is a DAX format option. If you attempt to format a DAX statement that is invalid, it will display a very lengthy complaint in the results window.

You can make things uppercase, lowercase, comment something out, uncomment something back in, and swap the delimiters to use either commas or semicolons. Merge XML is useful if you plan to use DAX parameters while querying, but it is not something I find myself using very often. Find and Replace are self-explanatory.

The Trace functionality is awesome and will be incredibly useful when you start to investigate why specific DAX queries might be taking longer than you'd expect. Is that visual taking longer than it used to? Use the Performance analyzer, get the query run by Power BI Desktop, copy it into DAX Studio, clear your in-memory cache, turn on Query Plan and Server Timings, and look to see if specific portions of the query are taking longer than others. Make sure you clear your cache before each run-through, as once the results have been run, that query pattern is placed into memory, which will speed it up.

You can see along the bottom, beneath the query results, a list of tabs showing Output, Results, Query History, All Queries, Query Plan, and Server Timings. The latter three tabs will appear only if you have the trace options selected in the ribbon.

All Queries cannot be enabled if Query Plan or Server Timings are enabled. All Queries can be useful when you want to know how many queries are being generated to get to a specific result.

My personal favorite function in DAX Studio is the Query Builder. In fact, the query shown in Figure 10-9 was generated by the Query Builder. I've been working on an NBA dataset recently and wanted to know, in this case, how many regular season games and how many points my favorite player of all time, Reggie Miller, scored in the regular season for each season he played.

You can see in Figure 10-10 that you can put columns and measures in the top portion of the Query Builder, and in the bottom half you can choose columns to filter by and the filter conditions. A section toward the bottom is for ordering the results.

As you bring in elements into the relevant areas, you can click Edit Query to generate the query in the Query window for you to view. This is a couple of great uses. Let's say you have a dataset in service, and an end user wants to extract a very specific collection of data. They could use DAX Studio, connect to the dataset in service, and use this Query Builder to get back the information they're looking for.

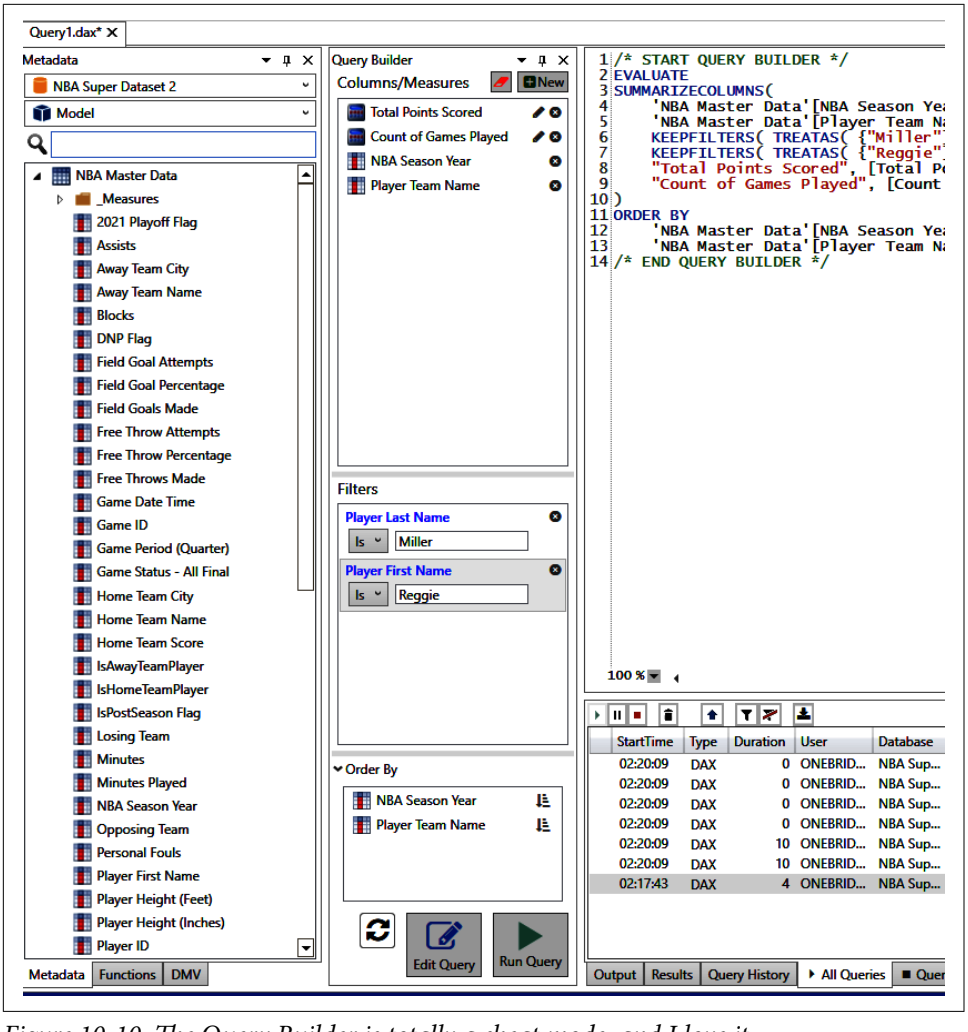


Figure 10-10. The Query Builder is totally a cheat mode, and I love it

Another cool use I've found is that this feature is great for DAX practice, especially with table functions. Often in Power BI we create measures designed to get specific values, but we don't necessarily work with the functions that return a table as its product. This gives us an opportunity to work with some of those functions in a way that doesn't impact our dataset itself. It also allows us to make manual edits and see what happens. It's a great learning tool in that regard.

Tabular Editor

Before I say anything about Tabular Editor, it's important to note there are two distinct versions of Tabular Editor, called Tabular Editor 2 and Tabular Editor 3. *Tabular Editor 2* is the original, open source solution that was developed. The same people who built Tabular Editor 2 built *Tabular Editor 3*, adding a lot of quality-of-life functionality, a nicer user interface, and some other things.

I'll be discussing Tabular Editor 2, which is the free, open source version. Currently there is no functionality in Tabular Editor 3 that can't be done in Tabular Editor 2, but it just might require some extra work.

As with DAX Studio, when we access Tabular Editor from the External Tools list, it will automatically connect to the Power BI data model that we have running. Tabular Editor is a no-frills, simple-to-use editor for any SSAS tabular model, just like the one that Power BI data models are built on top of. Let's take a look at the interface in Figure 10-11.

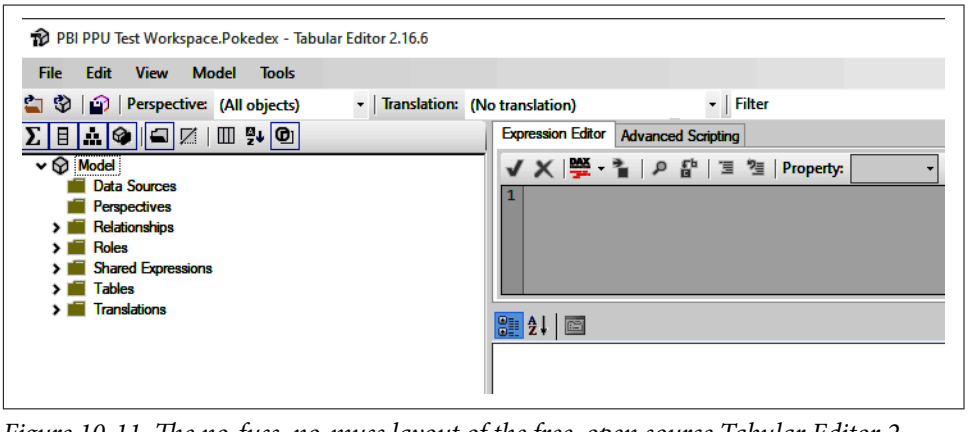


Figure 10-11. The no-fuss, no-muss layout of the free, open source Tabular Editor 2

At the top, we can see the classic Windows menu navigation: File, Edit, View, Model, and Tools. Beneath that we see three symbols. The first is an open folder that will allow you to open the files that Tabular Editor creates when you save a copy of the model metadata in Tabular Editor, which is a BIM file. These BIM files are basically a very large XML file, and some organizations manage their Power BI datasets entirely in Tabular Editor using BIM files. These can be source controlled, put into repositories, and are very small-sized files compared to their PBIX alternatives.

Next to that is an image of a transparent cube. This brings up the dialog box in Figure 10-12. In the Server section, we could put any analysis services database we have access to in the server address. This includes datasets currently in the Power BI service if you have XMLA read abilities enabled. Remember, this does require at least

Premium Per User licensing. If you have a Power BI dataset currently open in Power BI Desktop, you can choose one of those to connect to with the "Local instance" selector. In either case, you can choose whether you use Windows or Azure single-sign-on credentials or, if you need to pass a specific set of credentials, you can identify those credentials by choosing Username and Password and providing the appropriate information.

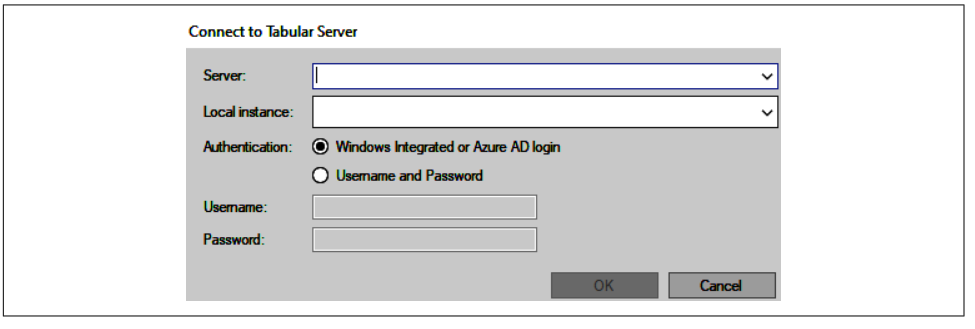


Figure 10-12. This dialog box pops up if you click the transparent cube in Tabular Editor 2. If you need to connect to a dataset in the Power BI service or an SSAS Tabular instance somewhere else, this can be a way to do it.

Finally in that group, we have what looks like a disk save icon. When this button is clicked, all the edits you made in Tabular Editor will be saved back to the database it's connected to. If that's your Power BI data model, all the measures you add, remove, or edit are going to get pushed all at once. You added some new relationships? Those will get added, too. Did you create new roles? You get the idea.

When that button gets clicked, the state of Tabular Editor will be pushed back to the system it's editing. Before you make changes to a Power BI dataset, whether that's locally or in service, make sure you have a backup. Have a Power BI Template (PBIT) file or a BIM file set to the side in case you make a mistake and need to roll back to a previous state because you modified something you didn't mean to change.

This does, however, come with an advantage. Let's say you want to create many measures for your data model. In Power BI Desktop, you must create them one at a time. Sometimes the interface is a little slow. You create the measure, but there's an error in the DAX, and you must go fix it. When the measure is ready, you want to add it to your display folder for organizational purposes, but you must do that one at a time too. That's a pain. Tabular Editor allows you to push multiple modifications to the data model at once. And while you're working on stuff, it doesn't affect the data model until the changes are saved.

Some things have a bit higher of a learning curve in Tabular Editor than they do in Power BI Desktop, though. As a result, I recommend a hybrid approach to using Tabular Editor until you feel more comfortable with the details that are needed to create many of these elements outside of Power BI Desktop.

On the Model Navigation pane on the left, you can see all the details of your model, and we can see that interface in Figure 10-13.

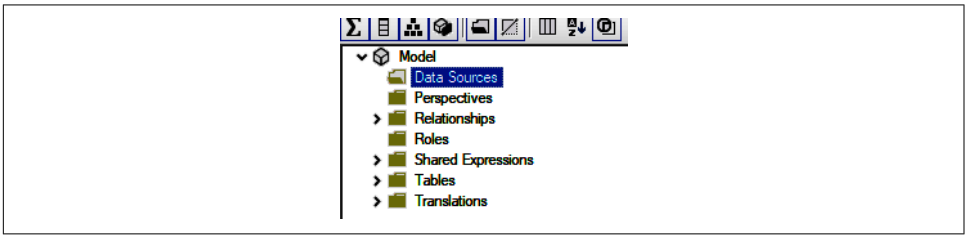


Figure 10-13. The underlying elements of the data model are shown in the Model Navigation pane

For any dataset that originates from Power BI Desktop, you won't be able to modify or even see the Data Sources. Perspectives have a very limited use case in Power BI, where they can be used with individual visuals if the Personalize Visuals option is turned on. Otherwise, they do nothing. You can create them, and they can exist in the data model, but Power BI doesn't support their functionality beyond the scope listed previously.

Basically, anytime you want to create a new element related to a portion of the data model, you right-click that element, and you'll see a contextual menu appear. I try to avoid managing relationships in Tabular Editor because I prefer a more visual way to see how all the tables fit together. However, you can create new relationships here if you like. If you're building a data model from scratch, you'll want to get used to this.

With Tabular Editor, we can also manage roles for RLS, and we can add measures, display folders, and other calculated items. Shared Expressions shows the list of what parameters exist in the dataset, and Translations shows what language or languages are supported by the dataset.

Creating Roles

Roles are what's required for RLS to work. We've talked about roles previously, and making them in Tabular Editor isn't terribly difficult. Using Tabular Editor also allows you to go beyond simple RLS and add object-level security (OLS) to a given role as well, controlling which objects in the data model a certain role can access.

In my Cool School University dataset, I have two roles that I made in Power BI Desktop and opened in Tabular Editor. In my NBA Master Dataset, I have two roles that I made completely outside of Power BI Desktop. They accomplish the same goal.

Let's look at my NBA dataset and use that to discuss making a role from scratch in Tabular Editor. I want to create a role that will show only the data for my favorite team, the Indiana Pacers, and one role that will show the data for everybody else. I'll right-click the Roles folder and select "new Role." The first thing I'm asked to do is to name the role. In this case, I'll call it "Pacers Only." Now, once that role is created, it doesn't do anything until it's defined. Just as we define roles using DAX in Power BI Desktop, we do that as well in Tabular Editor.

We can go look at the Properties pane in Figure 10-14 for the Pacers Only role. We care about the Security section and, specifically, the Row Level Security section. I want to add a DAX statement pertaining to a specific table—in this case, NBA Master Data. Given that this is player-level data, I want this role to display data only for players who were Pacers for the time they played for the franchise. The dataset in question has player information for each individual game, including their team affiliation, so I can define that as saying where the player's team name is the Pacers, and that will return data for players only for the time period in which they were members of the Indiana Pacers. That's easy enough to set up in DAX, and you can see where I've defined that.

Now, if I wanted to say that this Pacers Only role could see only certain tables, I could apply that at the table level here as well, in the OLS (Object Level Security) section, by changing the behavior for any table from Default to None. If I need column-level control for OLS, though, I need to go to the column specifically and change the role's OLS status to None.

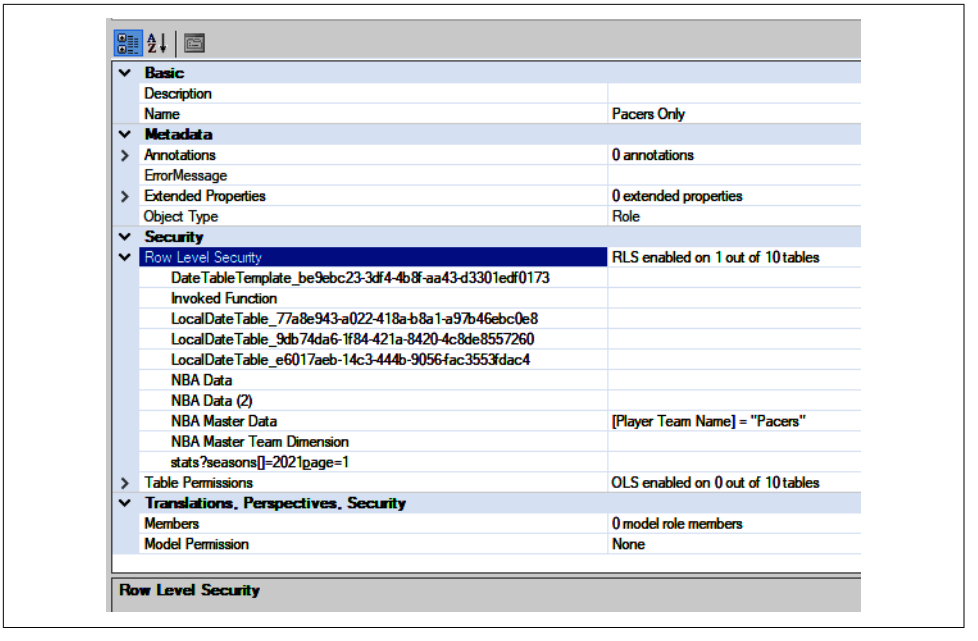


Figure 10-14. Implementing roles for RLS is pretty intuitive both in Tabular Editor and Power BI Desktop. Tabular Editor just gives you the extra power of OLS implementation as well!

Now that I have my first role created, knowing I want to create a role that is basically the same (except it's for a role that's all the teams except the Pacers), I can right-click my Pacers Only role, duplicate it, rename it, and then modify the RLS statement. In this case, I'll change [Player Team Name] = "Pacers" to [Player Team Name] = Not "Pacers."

Table and Measure Management

Here we get to how Tabular Editor can save every Power BI developer time and frustration: the creation and management of DAX-created elements. If you have calculated tables, calculated columns, or measures in your model, Tabular Editor can make your life better almost instantly. I'm going to focus on the Grades table from the Cool School University dataset. We can see this example in Figure 10-15.

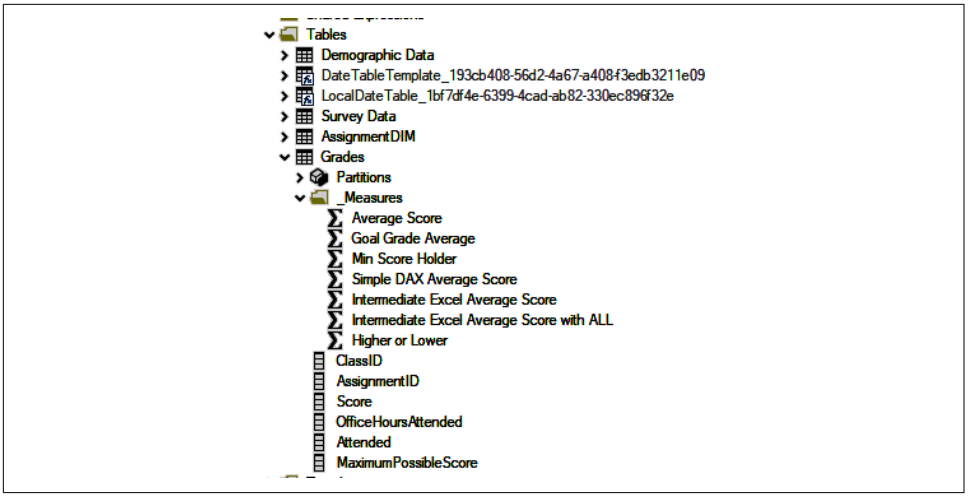


Figure 10-15. Tabular Editor can make seeing and managing your tables and measures in Power BI so much faster

If I right-click any table in my model, I can create a new measure, calculated column, hierarchy, data column, or data partition to split the table into multiple sections. If you have Incremental Refresh enabled, you'll see that Power BI takes care of creating the time-based partitions for you. I can choose to hide the table, duplicate it, or see the table's dependencies as well.

If I right-click a column, I can create a new display folder, measure, calculated column, hierarchy, or data column or set up a relationship from a given column to a different column. If you do manage relationships in Tabular Editor, this can also save you a ton of time.

You can also Ctrl+click to select multiple elements at once. Let's say you create 10 new measures using the expression editor on the right; you can deploy them all at once with the click of a button. You want to move a set of measures to your newly created display folder? You can do that. It's incredible how much faster Power BI Desktop seems to adopt those changes when they're pushed to it from Tabular Editor as opposed to when you do those things in Power BI Desktop itself.

Something Tabular Editor 2 doesn't do on its own, though, is version control, and it doesn't really keep a good history of changes you've made. This is critical because, no matter how good you get at this, you'll make a mistake sometime and need to either undo it or modify it. And you know that change could have happened multiple deployments ago. In this scenario, I would want to use a tool like the ALM Toolkit.

The ALM Toolkit for Power BI

The *ALM Toolkit* has a simple and useful purpose: identify what is different between two Power BI datasets. When the ALM Toolkit is chosen from Power BI Desktop, the source system will automatically choose the Power BI file you have open. If you aren't opening this from Power BI Desktop, you'll need to provide the source file (or the file in its current state) and the target file (the file that, in theory, you want to push the changes to). We can see that interface in Figure 10-16.

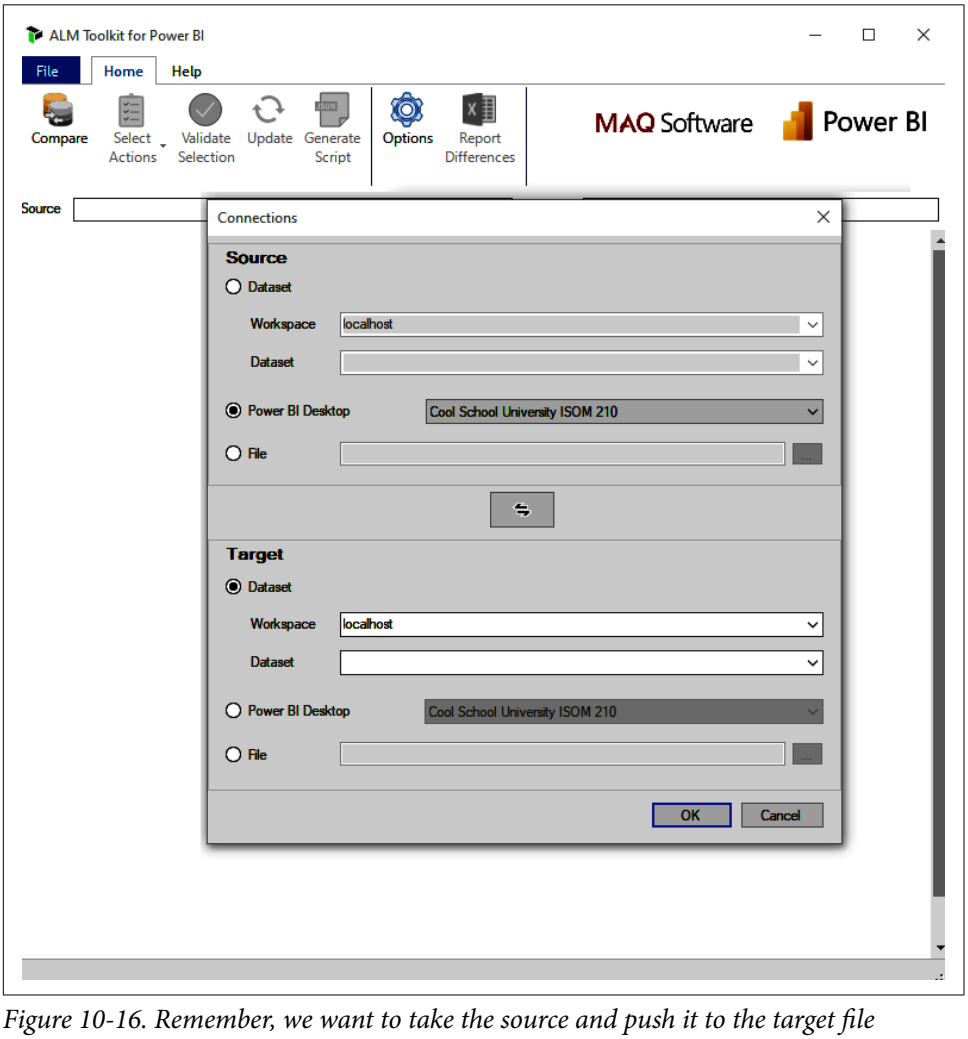


Figure 10-16. Remember, we want to take the source and push it to the target file

You'll notice a couple of options for where you can get the source and where you can point the target. The Dataset option allows you to point at a workspace in the Power

BI service, and, after you validate your credentials from a pop-up window, it will provide you with a list of all the datasets in that workspace.

Before deployment pipelines came about, it wasn't uncommon to have a development workspace with a development version of a dataset. You'd use the ALM Toolkit to push metadata changes from your development dataset in the Power BI service to a production dataset in the Power BI service in a totally different workspace.

The File option can be used only against PBIT files saved from Power BI Desktop or a BIM file from Tabular Editor. So, in this case, I'm going to compare my Cool School University ISOM 210 PBIX file to a PBIT file I made from this same dataset. I've made sure to make one change—in this case, removing a measure that was useless—so that we could easily see what changed and how that's displayed. You can see that in Figure 10-17.

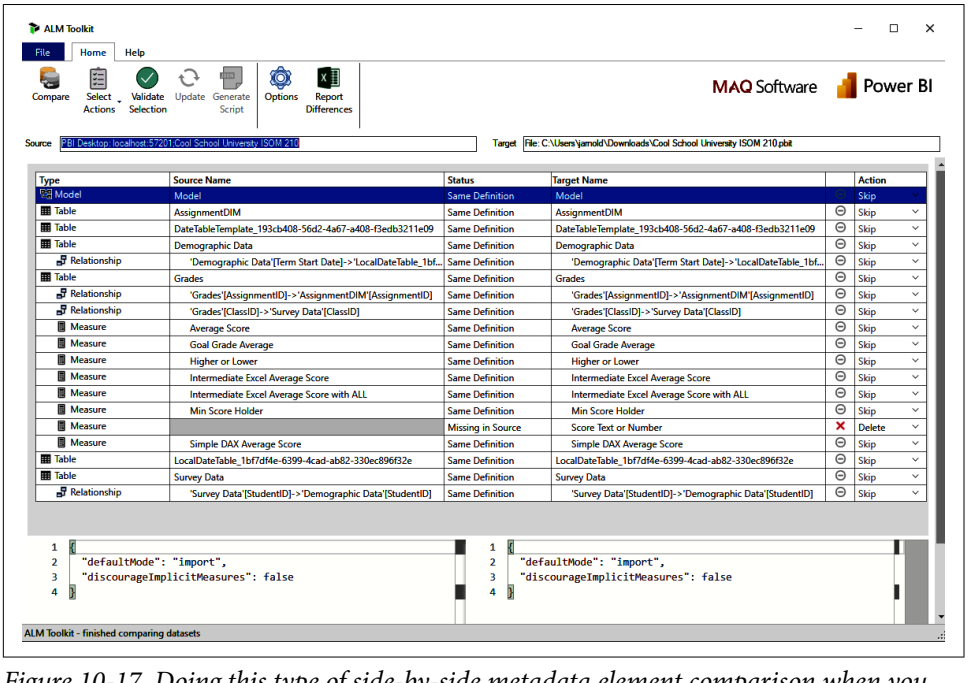


Figure 10-17. Doing this type of side-by-side metadata element comparison when you have two files is a lifesaver

I don't know about you, but I'm bad at remembering what I changed in a Power BI data model from day to day. Before I start doing any work on a Power BI Desktop file, I've gotten into the habit of saving a copy of that file as a PBIT. Now I have the PBIX I'm working with and a PBIT that contains all the metadata information. I'll make whatever changes I want to make (in this case, as shown in Figure 10-17), removing

the Score Text or Number measure, and confirm that I haven't made any other unintended changes.

You can see that the ALM Toolkit identifies changes across all the metadata in an Analysis Services database. You can also see what changed with a code-line comparison, like what you would see in, say, GitHub. Figure 10-18 shows what that looks like.

On the left, in the source system, you can see no lines of code because I removed them, and that shows as a delete action. On the right, however, you see, in JSON script, what was removed, including, importantly, the expression, which for a measure would be the DAX statement.

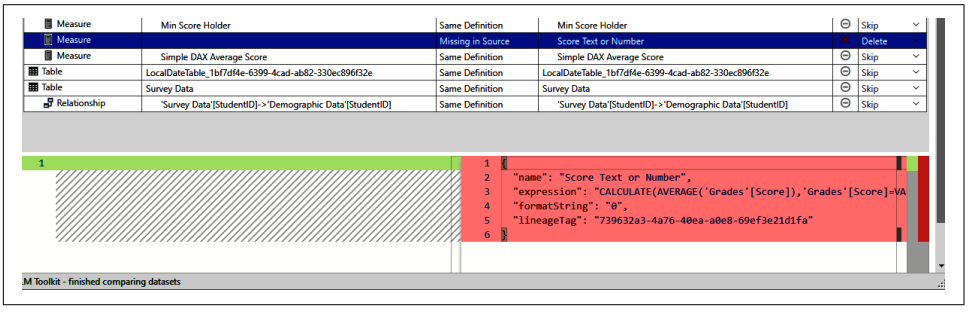


Figure 10-18. Code-level change detail management at your fingertips

In the Action section, I can choose whether I want to deploy that change or "skip" that change. A change that is skipped is ignored in the deployment of the metadata update from the source location to the target location. Maybe I changed a relationship to test how that would change a measure's behavior and forgot to change it back? This would catch that. I deleted an extra measure by accident, but I don't remember which one? This would catch that. I made an update to a calculated column and realize that the result is all wrong now, but I don't remember what the DAX was originally? This would catch that. I really can't express enough just how valuable this type of side-by-side metadata comparison is for your own sanity.

After you have the changes in place and ensured you have the list of changes you want to push, before choosing to validate the selection, make sure you capture the report differences. Report Differences will generate an Excel version of the table that you see in the software itself. This matters because this gives you an automatically generated change log. What does this allow us to do?

Let's say on April 24, I create the Excel file for the changes we saw earlier. I saved the Excel file with a date, so I know when the changes happened. Then I make some changes on May 1, June 1, and eventually it's September. In September, someone is trying to build a report on this dataset and comes to me and says, "Hey, wasn't there a measure for determining whether the score was in a text or number format around 61?" I can say that I removed it months ago, and when that person says that they need

it for something, I can go check my change log, find the original code, and reimplement it easily.

A change log can also be helpful if you are in a scenario with multiple developers and you might be making changes to a data model that could have impacts on other people's work. "Oh, hey, I made a change to calculations X, Y, and Z" doesn't have to put you in a tailspin of worry when you find out that change broke something. It's not a problem anymore because you can identify what changes occurred and isolate the change that broke the functionality you needed.

Every good piece of software has a change log when it gets updated. Treat your Power BI dataset like a good piece of software, and maintain a change log. Make it easier on yourself to do with the ALM Toolkit.

Once you have your log in place, choose Validate Selection to see a list of the changes it will push. Once you click OK, the Update and Generate Script buttons will become available. Update will push the change automatically, overwriting a new version of the file if it was a file that was the target location. Generate Script will generate an XMLA script that you can run in something like SQL Server Management Studio or another tool that will push that change. I recommend using the Generate Script method when you might have multiple people working with a given dataset and you want to confirm all the changes with other people. It can also be helpful if you want to keep a secondary log of the XML that was pushed as code to create the changes from the source to the target.

So we've talked about a tool to get tools, and a tool to modify our dataset, and a tool to do version control and change logging, but it would be nice to have a tool that would help us accelerate our development and give us high-level information about our dataset. This is where the newest tool in the third-party Power BI landscape comes in handy. That would be Bravo, from the folks over at SQLBI.

Bravo

Bravo is a user-friendly tool designed to help you get information to quickly optimize your model. If you open Bravo from Power BI Desktop, like all the other external tools we've seen, it will open the tool in the context of the current Power BI Desktop file being used. You can sign into the Power BI service from the tool and see information about datasets in Premium workspaces as well.

Analyze Model

The first thing you'll see in Bravo is the Analyze Model window, shown in Figure 10-19. We can use this to get very quick information about how large our dataset is, how many columns it has, and, more importantly, how many columns are not referenced within the model. It is important to know that Bravo cannot see if any reports that are dependent on the dataset might use one of these columns. But Bravo can help you figure out where to start looking if you can remove columns from your data model. This will help make the model both smaller and more efficient.

There's even a nice visual to show how much of the model is taken up by specific columns or, in my example, a collection of smaller columns. If I click the smaller columns collection in the columns list, the visual will break out those columns as well. Anything highlighted in yellow is currently not being used in the model and may be a candidate for removal. There's a search option and a filter option, both of which can be helpful if you have a model with many columns. There is also an option to download a VPAX file, which is a file used in software called VertiPaq Analyzer that is also made by the folks at SQLBI. I like to use Bravo to identify columns that I can remove, and I'll keep a record of those column changes using the ALM Toolkit to deploy those changes to an updated PBIT or BIM file when I'm ready to save my changes.

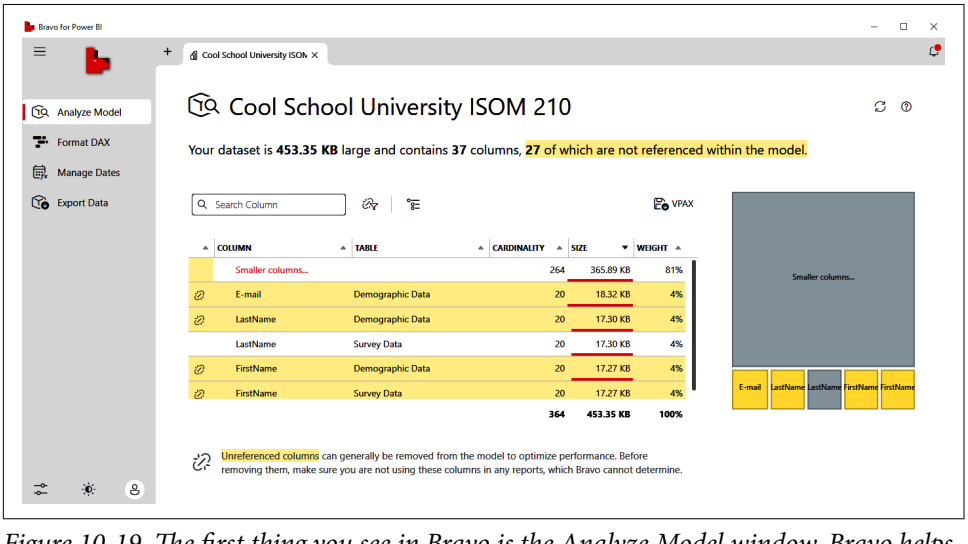


Figure 10-19. The first thing you see in Bravo is the Analyze Model window. Bravo helps make your model healthier and easier to understand, and solves some real annoyances.

DAX Formatting

If you get Bravo for nothing else, the Format DAX page is an absolute godsend. You can see all your measures, and by clicking Analyze Now, that script will be read by the DAX Formatting service. It will tell you how many of your measures have errors and how many measures you have that aren't formatted as the DAX Formatting service suggests. You can choose individual measures, or you can select all and have them mass formatted. If you click a measure, a window to the right appears that will show you both the current format and a preview of the formatted DAX. If you have a reason that you have a specific measure formatted a certain way, you can leave that one alone and fix the others. With something like Bravo, there's no reason to have unformatted DAX, and when your DAX is formatted and someone else's isn't, I promise you, you'll look better.

Manage Dates

Bravo can create a date table for you and create a ton of time intelligence measures based on measures you already have in your model. This feature can save you so much time as dozens of properly formatted measures get created to accelerate your development. However, there is a catch. First, you cannot have auto date/time enabled in your Power BI data model and, second, you cannot have another table already identified as your date table. Bravo will tell you if you can use the Manage Dates features with your current data model. If you can, then you can quickly create a date table with set time intervals, your language of choice, even which country's holidays to add to the model. I do hope in the future they will add the ability to have multiple countries' holidays.

If that was all this thing did, that would be enough to be awesome. It gets better. The time intelligence section will ask if you want these measures to be enabled and, if so, do you want time intelligence built for all of your measures or a subset that you choose? In Figure 10-20 you can see a quick example of how deep the measure rabbit hole goes in Bravo. It will also create the display folders for you, which is incredibly useful. In addition, if you want to get better at DAX, there are some wonderful examples of how to write DAX using time intelligence functions that you can use to help push you along in your DAX journey.

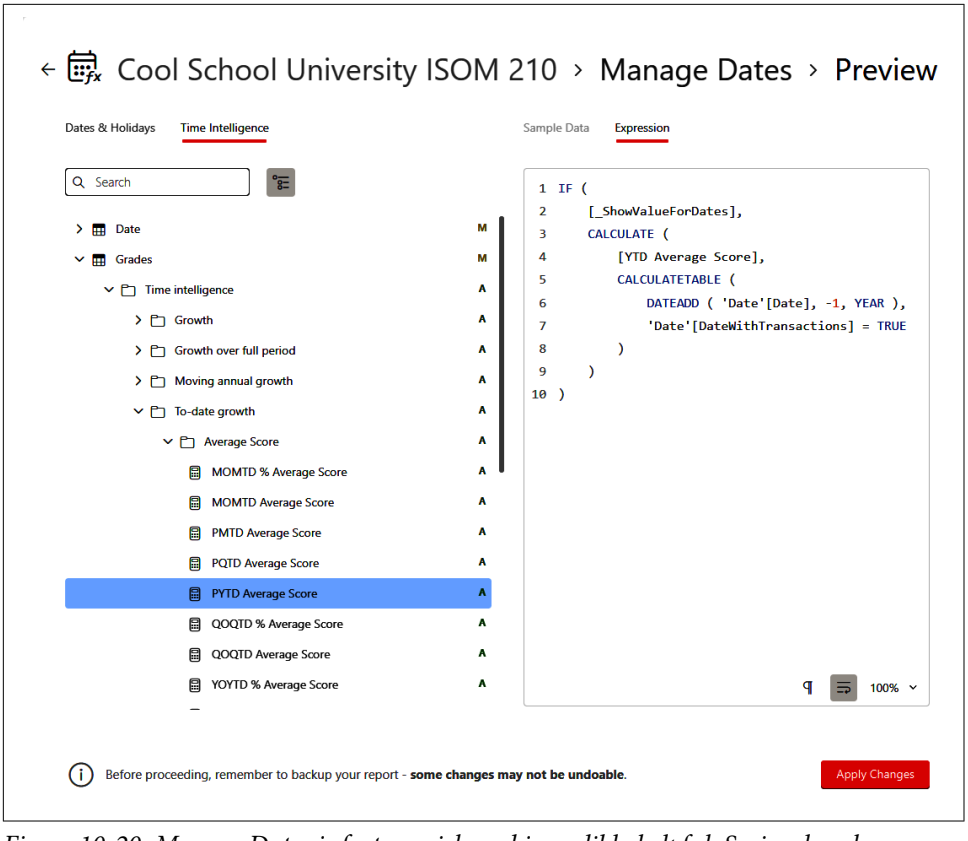


Figure 10-20. Manage Dates is feature-rich and incredibly helpful. Seriously, where was this when I was learning Power BI?

Export Data

The last thing that Bravo can do is provide an export of data from a table in a data model. It's as simple as clicking the box or boxes you want for the tables you want to export and choosing whether you want to export that data as an Excel spreadsheet or a CSV file. Bravo can also provide you with an export summary page if you want additional details around what data was exported. When the export is complete and the file is saved, Bravo even visually gives you a link to click to take you straight to the file.

Now, it is still exporting to either Excel or CSV, so it can't export an infinite number of rows. If you have a process that is requiring you to get millions of records from a table, you're probably better off using an Evaluate statement in DAX Studio and exporting the results from there. However, if your table isn't millions of rows deep, Export Data can be a really great way to export data from a table in your data model.

Conclusion

Well, we've reached the end of the road. We started with zero knowledge of Power BI and went all the way through learning how to use Power BI Desktop and the Power BI service. Then we ended here, with tools to accelerate and empower your Power BI development.

The Power BI ecosystem is wide and deep. Microsoft is going to continue to push the Power Platform forward, and Power BI will be the data cornerstone of that process. If you've read through this book and feel confident enough now to build that report, to bring together that data, to help a user get access to a workspace, or to set up a robust development pipeline, then I feel like I've accomplished something.

And you should feel like a proud graduate of Cool School University. You've worked hard, put in your time, and now you've elevated your skill set. With this, you can impress your colleagues and make an impact on your organization. Knowing Power BI will enable you to provide data analytics in a visual output that technical folks and business users alike can understand and make smarter decisions with.

I can't say I know where Power BI will be going in the future, but I know I'm excited to see it. I hope you're now ready to take on whatever data challenge lies ahead. A wise man in exile once told me that the end of learning is the beginning of death. Data is the oxygen that allows us to learn in the 21st century. Never stop learning. Good luck and Godspeed.

APPENDIX A

Commonly Used DAX Expressions

In this appendix, I will go over the syntax of some of the most commonly used DAX functions in Power BI. These are organized by section, then alphabetically within that section. For each function, there will be a brief description of the purpose of the function, the syntax of the function, and then an example using that syntax.

This is not an exhaustive list of all the functions in the DAX language. That can be reviewed at "DAX Function Reference".

As with many things, DAX is a prime example of the Pareto principle: 80% of all the problems (outputs) can be solved with 20% mastery (inputs).

Aggregation Functions

AVERAGE

Definition

Returns the average (mean) of all the numerical values of a column.

Syntax

```
AVERAGE ( [ColumnName] )
```

Example

```
AverageScore = AVERAGE ( GradeScore[Score] )
```

AVERAGEX

Definition

Calculates the average (mean) of a set of expressions evaluated over each row of a table.

Syntax

```
AVERAGEX ( 'TableName' , <expression> )
```

Example

```
AverageScorePercentage = AVERAGEX ( 'GradeScores', 'GradeScores'[Score] +
  'GradeScores'[MaximumPossibleScore] )
```

COUNT

Definition

Returns the number of records for a column that are not blank.

Syntax

```
COUNT ( [ColumnName] )
```

Example

```
CountOfStudents = COUNT ( 'UniversitySuppliedData'[StudentID] )
```

DISTINCTCOUNT

Definition

Returns the number of distinct values for a given column.

Syntax

```
DISTINCTCOUNT ( [ColumnName] )
```

Example

```
CountOfStudents = DISTINCTCOUNT ( 'GradeScores'[StudentID] )
```

MAX

Definition

Returns the largest value in a column.

Syntax

```
MAX ( [ColumnName] )
```

Example

```
HighestScore = MAX ( 'GradeScores'[Score] )
```

MAXX

Definition

Returns the largest value for an expression over each row of a given table.

Syntax

```
MAXX ( 'TableName', <expression> )
```

Example

```
LargestScoreAndOfficeHours = MAXX ( 'GradeScores', 'GradeScores'[Score] +
  'GradeScores'[OfficeHoursAttended] )
```

MIN

Definition

Returns the smallest value in a column.

Syntax

```
MIN ( [ColumnName] )
```

Example

```
LowestScore = MIN ( 'GradeScores'[Score] )
```

MINX

Definition

Returns the smallest value for an expression over each row of a given table.

Syntax

```
MINX ( 'TableName', <expression> )
```

Example

```
LowestScoreAndOfficeHours = MINX ( 'GradeScores', 'GradeScores'[Score] +
  'GradeScores'[OfficeHoursAttended] )
```

SUM

Definition

Adds all the numbers in a given column.

Syntax

```
SUM ( [ColumnName] )
```

Example

```
TotalOfficeHoursAttended = SUM ( 'GradeScores'[OfficeHoursAttended] )
```

SUMX

Definition

Returns the sum value for an expression over each row of a given table.

Syntax

```
SUMX ( 'TableName', <expression>)
```

Example

```
TotalEffectiveScore = SUMX ('GradeScores', 'GradeScores'[Score] +
  ('GradeScores'[OfficeHoursAttended] * 20) )
```

PRODUCT

Definition

Returns the product (multiplication) of the numbers of a column.

Syntax

```
PRODUCT ( [ColumnName] )
```

Example

```
ScoreMultiplication = PRODUCT ( 'GradeScores'[Score] )
```

PRODUCTX

Definition

Returns the product (multiplication) of an expression evaluated for each row in a table.

Syntax

```
PRODUCTX ( 'TableName', <expression> )
```

Example

```
BonusScoreMultiplication = PRODUCTX ( 'GradeScores', 20 *
  'GradeScores'[OfficeHoursAttended] )
```

Date and Time Functions

CALENDAR

Definition

Returns a table with a single-date column that is a continuous list of all dates between the start date and the end date. This can accept DAX statements that would result with a date.

Syntax

```
CALENDAR ( <start_date> , <end_date>)
```

Example

```
DateRange = CALENDAR ( "01/01/2022" , "12/31/2022" )
```

DATEDIFF

Definition

Returns the count of a given set of intervals between two dates, where an interval can be SECOND, MINUTE, HOUR, DAY, WEEK, MONTH, QUARTER, or YEAR.

Syntax

```
DATEDIFF ( <start_date>, <end_date>, <interval>)
```

Example

```
DaysBetweenTerms = DATEDIFF ( MIN('UniversitySuppliedData'[Term Start Date]),
  MAX('UniversitySuppliedData'[Term Start Date]), DAY)
```

DAY

Definition

Returns the day of the month of a given date between 1 and 31.

Syntax

```
DAY ( <date> )
```

Example

```
DayOfTermStart = DAY ('UniversitySuppliedData'[Term Start Date])
```

MONTH

Definition

Returns the month of a given date between 1 and 12.

Syntax

```
MONTH ( <date> )
```

Example

```
MonthOfTermStart = MONTH ('UniversitySuppliedData'[Term Start Date])
```

TODAY

Definition

Returns the current date.

Syntax

```
TODAY()
```

Example

```
DaysSinceMostRecentTermStart = DATEDIFF ( Today(),
  MAX('UniversitySuppliedData'[Term Start Date]), DAY)
```

YEAR

Definition

Returns the year of a given date.

Syntax

```
YEAR ( <date> )
```

Example

```
YearOfTermStart = YEAR ('UniversitySuppliedData'[Term Start Date])
```

Time Intelligence Functions

DATEADD

Definition

Returns a table that contains a column of dates, either in the future or in the past by the specified number of intervals from the initial date.

Syntax

```
DATEADD ( <dates>, <number_of_intervals>, <interval> )
```

Example

```
ThirtyDaysFromTermStart = DATEADD ( 'UniversitySuppliedData'[Term Start Date],
  30, DAY )
```

DATESMTD

Definition

Returns a table that contains a column of dates in the current month to date.

Syntax

```
DATESMTD ( <dates> )
```

Example

```
MTDDatesOfTermStart = DATESMTD ( 'UniversitySuppliedData'[Term Start Date] )
```

DATESQTD

Definition

Returns a table that contains a column of dates in the current quarter to date.

Syntax

```
DATESQTD ( <dates> )
```

Example

```
QTDDatesOfTermStart = DATESQTD ( 'UniversitySuppliedData'[Term Start Date] )
```

DATESYTD

Definition

Returns a table that contains a column of dates in the current year to date.

Syntax

```
DATESYTD ( <dates> )
```

Example

```
YTDDatesOfTermStart = DATESYTD ( 'UniversitySuppliedData'[Term Start Date] )
```

Filter Functions

ALL

Definition

Returns all the rows in a table or column, ignoring all filter context for the selected table and column. Note that you can choose to not specify a column and use only the TableName portion of the function, effectively clearing filters on all columns for the chosen table. The following example demonstrates this. If you wanted to add specific columns to the ALL statement, refer to the full syntax, here.

Syntax

```
ALL ( ['TableName', [ColumnName1], [ColumnName2],...,[ColumnNameX] )
```

Example

```
%OfTotal = SUM('GradeScores'[Score]) / CALCULATE(SUM('GradeScores'[Score]),
  ALL('UniversitySuppliedData'))
```

ALLEXCEPT

Definition

Removes all filter context except for those columns specifically selected in the ALL EXCEPT statement.

Syntax

```
ALLEXCEPT ('TableName'[ColumnName1],[ColumnName2],...,[ColumnNameX])
```

Example

```
ScoreTotalOrByLastName = CALCULATE(SUM('GradeScores'[Score]),
  ALLEXCEPT(UniversitySuppliedData,UniversitySuppliedData[LastName]))
```

CALCULATE

Definition

Evaluates an expression with modified filter context. Remember, you will use this one a ton.

Syntax

```
CALCULATE(<expression>, {FilterCondition1}, {FilterCondition2}
  ,..., {FilterConditionX})
```

Example

```
AverageScoreAssignment1 = CALCULATE(AVERAGE('GradeScores'[Score]),
  'AssignmentDIM'[AssignmentID]=1)
```

Logical Functions

AND

Definition

Performs a test to determine whether both of the passed arguments are true. This function returns `TRUE` if both arguments pass, and otherwise returns `FALSE`.

Syntax

```
AND ( <logicalcondition1>, <logicalcondition2> )
```

Example

```
SpecificKingScoreForComparison = CALCULATE(sum('GradeScores'[Score]),AND
  ('UniversitySuppliedData'[LastName] = "King",
  'UniversitySuppliedData'[FirstName] = "Leonard"))
```

COALESCE

Definition

Returns the first expression that doesn't come back BLANK. If everything returns blank, BLANK is returned.

Syntax

```
COALESCE ( <expression1>, <expression2>,..., <expressionX)
```

Example

```
TotalScoreTreatingBlanksAsZeroes = COALESCE ( SUM ( 'GradeScores'[Score]), 0)
```

IF

Definition

Checks a given condition, returning one result if TRUE and going to a second result if FALSE.

Syntax

```
IF (<logical test>, <value_true>, <value_false>)
```

Example

```
CurveEligible = IF ('UniversitySuppliedData'[YearsAttended] <=2, "Yes", "No")
```

OR

Definition

Performs a test to determine whether either of the passed arguments is true. This function returns TRUE if either argument passes and otherwise returns FALSE.

Syntax

```
OR ( <logicalcondition1>, <logicalcondition2> )
```

Example

```
CountOfIndianaOrHawaiiStudents = CALCULATE(COUNT
  (UniversitySuppliedData[StudentID]),
  OR(UniversitySuppliedData[ResidencyState]="IN",
  UniversitySuppliedData[ResidencyState]="HI"))
```

DAX Operators

DAX has many operators that perform functions similar to what you would expect, but some have unique functionality. Table A-1 lists operator names, operator signs, and their function.

Table A-1. DAX operators

Operator name	Operator sign	Function
Plus sign	+	Addition
Minus sign	–	Subtraction
Asterisk	*	Multiplication
Forward slash	/	Division
Caret	^	Exponential
Equal sign	=	Equal to
Double equal sign	==	Strictly equal to
Left caret	>	Greater than
Right caret	<	Less than
Right caret and equal sign	>=	Greater than or equal to
Left caret and equal sign	<=	Less than or equal to
Left caret right caret	<>	Not equal to
Single ampersand	&	Text concatenation
Double ampersand	&&	Logical AND operator
Double pipe	||	Logical OR operator
IN	IN	Logical OR condition for a given list of values

APPENDIX B

Some Favorite Custom Visuals

An incredible amount of development effort has been put into the creation of custom visuals in Power BI. In fact, in my last cursory count of custom visuals in Power BI's AppSource listing, I found 404! Many of the custom visuals in this list are 100% free to download and use. Some have free elements and then elements you can license to expand their functionality; for example, the xViz, ZoomCharts, and Zebra BI visuals work this way. Everything in the app store, to my knowledge, has at least some free components.

Something to think about when you use custom visuals is whether a visual is certified. Certified visuals have extra functionality in line with the prepackaged visuals from Power BI, including the ability to export their results to PowerPoint or embed the visual in emails from subscriptions. Just because a visual is not certified does not mean it's unsafe or a security risk. For more information on custom visual certification, see Microsoft's documentation on the matter at "Get Your Power BI Visual Certified".

In this appendix, I'll provide a quick review on how to add custom visuals. I'll also review some of my personal favorite custom visuals. Finally, I'll introduce Charticulator, a free-to-use, no-code-required custom visual builder offered from Microsoft. For clarity, I will not include any visual that has a licensing component because, in full disclosure, I personally haven't used a lot of them and can't really speak to their prospective advantages over existing free options.

Adding Custom Visuals to Power BI Desktop

Adding a custom visual is a painless process if you know where to look. In the Visualizations pane alongside all the default Power BI visuals, look for an ellipsis after the last visual. Clicking this brings up a small menu that provides options to get more

visuals, import a visual from a file, remove a visual, or restore default visuals. Before AppSource was as prevalent in the Microsoft ecosystem as it is today, many custom visuals were created as PBVIZ files. If you have a PBVIZ either from a custom visual you've created using TypeScript or an older version of a custom visual that you want to import from a file, you can select that option to access an Explorer window to navigate to the file and select it for importing.

If you select the "Get more visuals" option, an overlay will appear that will bring you to a list of custom visuals that are available in AppSource and custom visuals that have been added to your organization. Items from your organization will always appear before nonorganizational items in the list of all visuals. We can see what this overlay looks like in Figure B-1.

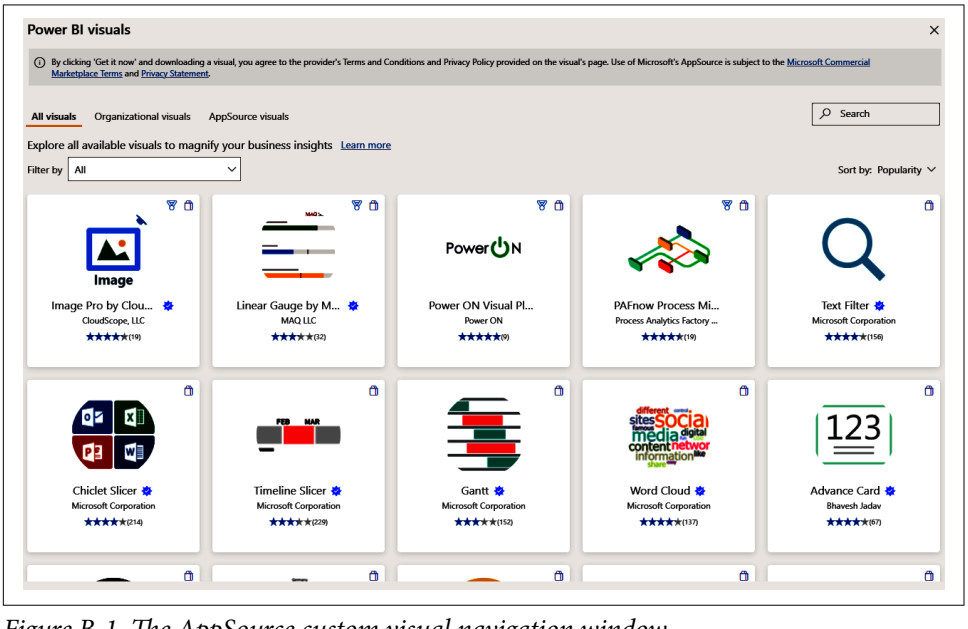


Figure B-1. The AppSource custom visual navigation window

You can see there are selection options for all visuals, organizational visuals, and AppSource-only visuals. There's search functionality. There is also the ability to filter the visuals categorically. These categories are Analytics, Advanced Analytics, Change Over Time, Filters, Infographics, KPI, and Maps. The filter categories aren't always helpful in defining what you're looking for, and this is particularly true in the Analytics category, which I find overly broad. However, you can use both filter and search together, which you might find more helpful.

Something else that is helpful in using each of these custom visuals from AppSource is that they all have a sample PBIX that you can use to see how they work and what they do. This provides a very low barrier to entry, as you can see the visual at its best,

the defined use case from the creator of the visual, and decide if it is something that might work for you.

Ten of My Favorite Custom Visuals

I said there were 404 custom visuals in AppSource based on my last count. With that many, I certainly haven't had a chance to use them all, and frankly I don't know anyone who has. However, I keep coming back to certain visuals for different purposes time and time again, so I've composed a list of 10 that are either broadly applicable or that really do a good job of solving the problem they are designed to address. I'll provide you with a brief overview of each of them. I could've made this list two or three times as long, but I chose the ones I think are most helpful. Please note that in many cases, I'll demonstrate the use of these visuals by using the provided sample dataset.

- Advance Card
- Chiclet Slicer
- Drilldown Choropleth
- Forecasting with ARIMA
- Gantt chart
- Radar Chart
- Route map
- Scroller
- Sunburst chart
- Word Cloud

Advance Card

The *Advance Card visual* is not a radical overhaul of the basic card visual, but a straightforward enhancement. Advance Card supports additional features such as conditional formatting, conditional formatting based on another measure, easy text alignment, tooltip support, and background image support. We can see some of what that looks like from the example PBIX for the visual in Figure B-2.

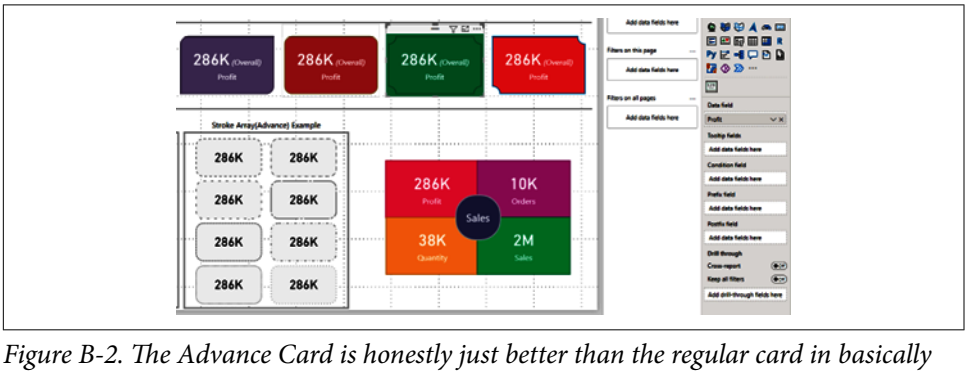


Figure B-2. The Advance Card is honestly just better than the regular card in basically every way

Chiclet Slicer

I love the *Chiclet Slicer*, which is a slicer that has controls for the number of rows and columns of values you want to condense your selections into. I personally just love the "button" feel of the Chiclet Slicer.

Another cool function of the Chiclet Slicer is that you can associate images with different "chiclets." A common example of this is a report about multiple countries, and each chiclet might display a national flag. Even without the additional chiclet formatting, I really like how the Chiclet Slicer looks and behaves.

Microsoft's initial announcement of the Chiclet Slicer back in 2015 used car company logos as an example. I like to store images for Chiclet Slicers in a OneDrive folder or SharePoint site and then have a column in my data model that calls those links as Image URLs. Figure B-3 shows Microsoft's car example of the Chiclet Slicer, alongside the visualization parameters.

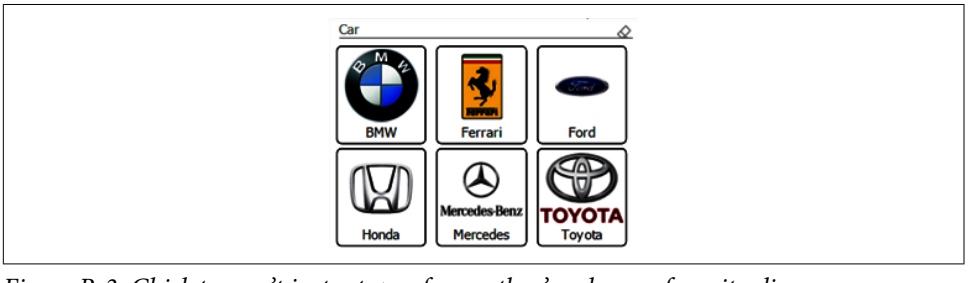


Figure B-3. Chiclets aren't just a type of gum, they're also my favorite slicer

Drilldown Choropleth

I feel like I need to provide a warning first on this visual. It is not user-friendly to set up. You need to have TopoJSON files to identify how your maps should be shaped. Unlike the shape map visual, you will provide a link to the TopoJSON files you will use for your geographic drilldown in the visual's formatting pane. If you can't find a TopoJSON of the geographic boundaries you are looking for, you might be able to find it as a shapefile or GeoJSON, both of which have free online tools that will allow you to convert one type of file to another.

A *choropleth* is a map that uses color to show differences between geographic regions. The reason I like the *Drilldown Choropleth visual* is that even with its difficult setup, once you have everything working, you can allow users to go easily from one geographic level to another and keep all the context in a single visual. As you can see in Figure B-4, I go from the state/territory level of Australia down to the postal code level after going through the state-level filter to get to Western Australia, and my results remain contextual.

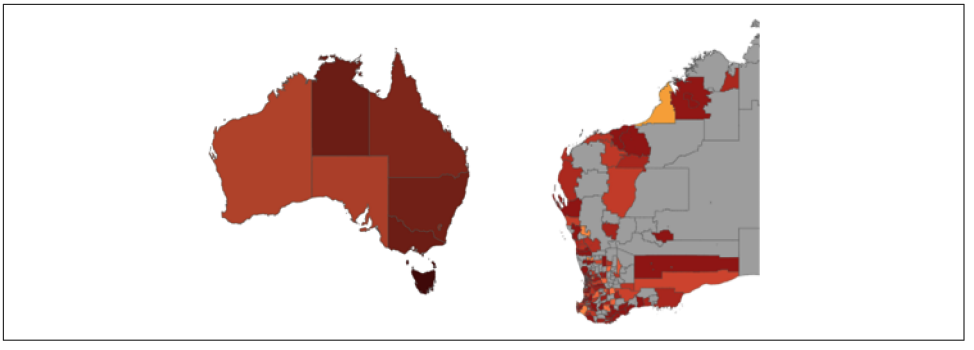


Figure B-4. TopoJSONs are powerful mapping tools, even if they're a pain to set up

Forecasting with ARIMA

Two of my favorite "simple" forecasting methods are exponential smoothing and Autoregressive Integrated Moving Average (ARIMA). It's important to note that this is an R-powered visual, which means to use it you will need to have R installed. The visual will prompt you to install the correct libraries when you first add ARIMA. The downside is that to refresh a data model that uses this visual, you will have to use a personal data gateway.

However, I like this visual because it is simple and provides easy-to-understand confidence intervals for the results. It also has several controls for more seasoned analysts to fine-tune the forecast to account for specific scenarios or parameters. For a more detailed explanation on ARIMA and its use in Power BI, I suggest downloading the sample dataset for this visual, as its report pages do a good job of explaining the ARIMA methodology. As a bonus, I think the visual actually looks pretty nice in its presentation. We can see what that looks like alongside its formatting options in Figure B-5.

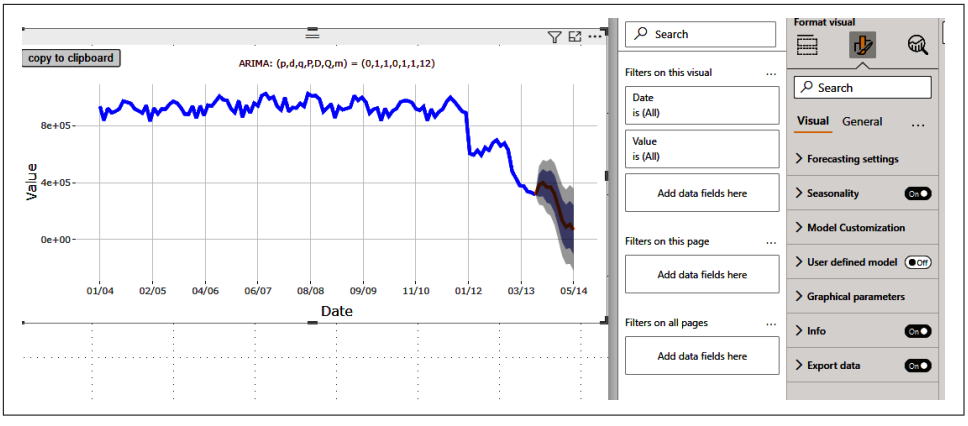


Figure B-5. ARIMA is not a universal bullet, but it provides a great starting point in your prospective forecasting

Gantt Chart

The *Gantt chart* has been around forever, and it's a staple of project management. It's a type of bar chart with a time series that helps you identify events in terms of estimated time to completion and the number of steps involved.

AppSource has a couple of Gantt chart options, but in this example, I'm specifically talking about the version from Microsoft. What I really like about this version is that it can be as broad or as detailed as you want to make it. This Gantt chart has 10 field selections you can add values to. The only field that is required is the task field.

The visual sample PBIX has an easy-to-read hints page that I often reference when I need to remember which field does what, since I'm not in project management that often. Honestly, I wish some of the formatting options in this chart were available in some base visuals, but you can also use the Gantt chart in "off-label" use cases to borrow some of that functionality for more classic bar chart use cases. We can see an example of the Gantt chart visual in Figure B-6.

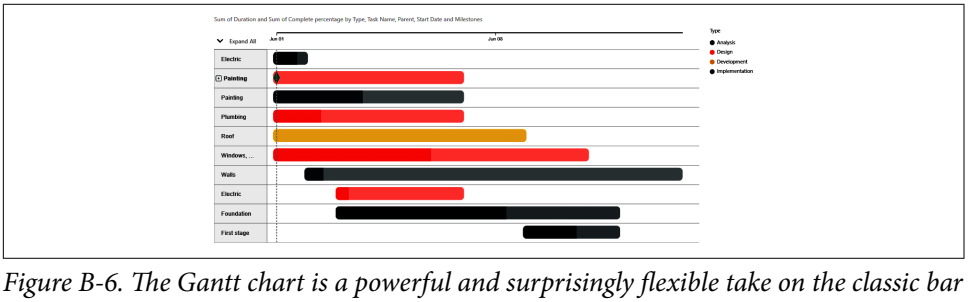


Figure B-6. The Gantt chart is a powerful and surprisingly flexible take on the classic bar chart

Radar Chart

The *Radar Chart* is a visual that allows you to display multiple variables' worth of data to see a broader holistic picture across those variables. Generally speaking, the axes should not be considered comparative, as different variables will have different scales on their respective axis. For instance, the example in Figure B-7 shows the count of new employees by division versus the forecast number of new employees by division. We don't necessarily know what the absolute values are, but we can quickly compare how each group did with actuals to forecast in a broad interpretive sense.

One of my favorite examples of the use of the radar chart, not in Power BI, is in Street Fighter 5, where each character has a radar chart that shows their "stats" out of a rank of 5, quickly allowing you to see which characters have which general strengths. Want a character with a lot of life that hits hard or a quick character with many special moves? The radar chart works perfectly in that context as well, showing it's a chart type with broad applications. It's also easy to use in Power BI since it asks only for a category list and y-axis values.

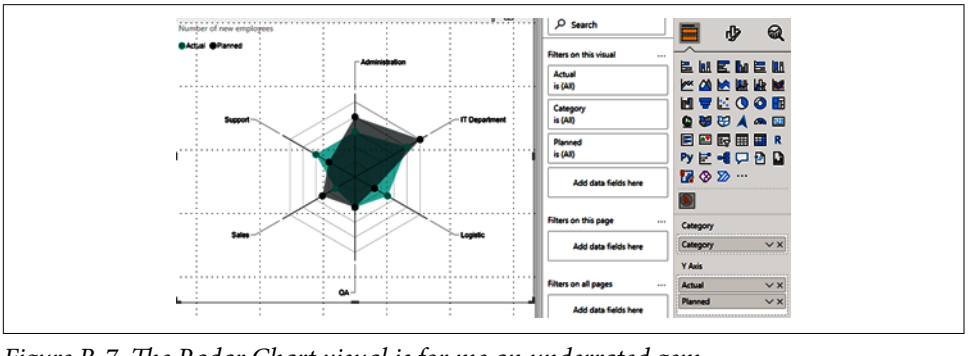


Figure B-7. The Radar Chart visual is for me an underrated gem

Route Map

The *"Route map" visual* helps visualize data that shows geolocation at different points across an axis, usually time. So, for example, if you were a trucking company and wanted to see the route your trucks were taking, you could get data from their GPS, model that data, and use this visual to note those routes and possibly run some optimization analysis.

My wife uses this visual with her geolocation data to map her runs. The visual does pass that geolocation data to a third-party service, which means it will not ever be certified, so that is something to keep in mind. You can see, though, that the map it produces in Figure B-8, from OpenStreetMap, is very readable.

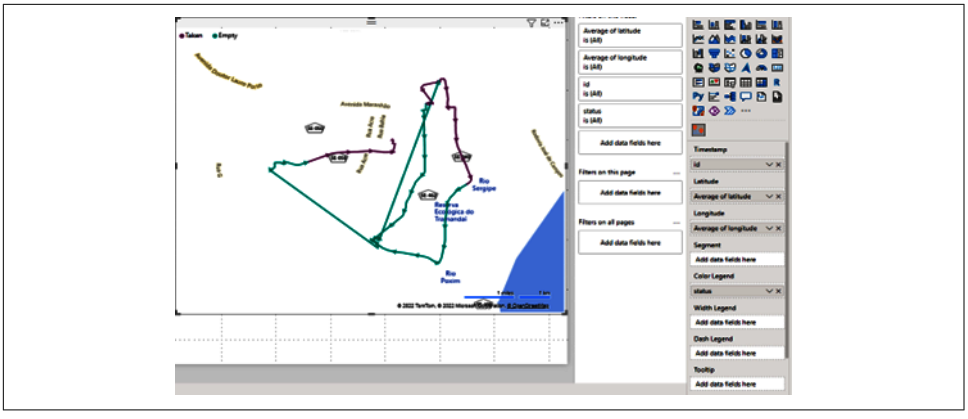


Figure B-8. The Route map is a visual that can really spice up geolocation data

Scroller

The *Scroller visual* is your classic ticker. It showcases data in categorical context with the option of adding a change measure. I like to use the Scroller visual to compare actuals with forecast values or against a previous day's individual value (say, product sales or shipments). I also like to use the Scroller visual when I have a report page that will be displayed and visible (say, a manufacturing floor). The Scroller visual can be used to provide repeating messages or other additional notices.

I had one customer who used it on a television display to remind employees of upcoming events, holiday notices, birthdays, anniversaries, etc. Everyone has probably seen a scrolling ticker at some point in their life, but the real power of having control over the data you put with this visual gives it so much flexibility. This implementation has controls over scroll speed, font size, and the ability to add color to status indicators and status text.

Sunburst Chart

A *Sunburst chart* is a circular chart (like a pie or a donut) with multiple "rings" that allow for different categorical combinations to identify an aggregation at any or all levels of the categorical split. Even though I generally don't like pie or donut charts, I like that for any given combination I select from the Sunburst, I get to see a specific value in the middle of the Sunburst that gives me numerical context to the combinations I'm looking at without having to guess. With multiple categorical options in the Sunburst chart, you can also use this to get to specific combinations of categories to filter other visuals on the page.

In Figure B-9, I have the Category 2 / SC2 / Asia grouping selected, and those values are passed to all the other visuals on the page from a cross-filtering perspective at once, as opposed to having to do three separate sets of cross-filtering. I also can

choose any combination of those levels. All of Category 2? I can do that. Category 1 and SC1? I can do that. Category 2 / SC1 / North America? Yep. So it does have some pie and donut baggage, but it does enough differently that I think it's worth including in this list.

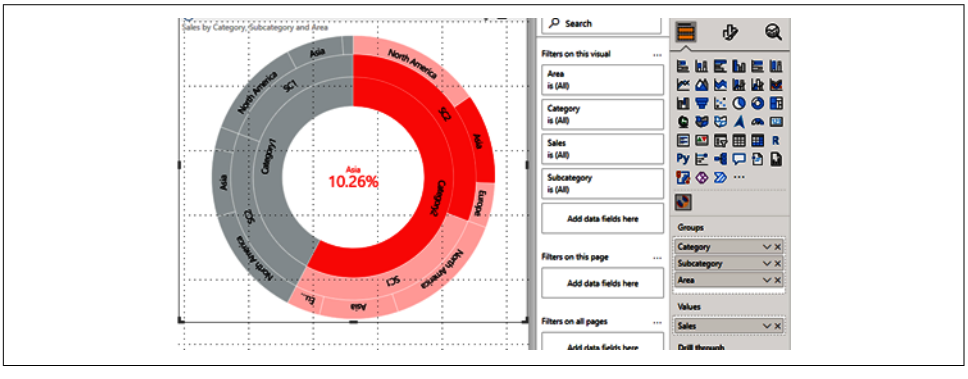


Figure B-9. The Sunburst chart, my hypocritical indulgence

Word Cloud

The *Word Cloud* is a visualization that we are seeing used more and more in a variety of contexts. A word cloud takes words and identifies how often they show up. I tend to use this visual most often in marketing analytics where I am trying to get some insight into consumer sentiment or understand what topics are dominating a conversation. Let's say we have a focus group and we're breaking down what the group members said about a product; I'd want to know the most common words used to describe my product.

This also comes up quite a bit in survey data. The count of how often a word was selected could also easily be visualized in a word cloud. This visual in its formatting also allows for you to choose words to exclude and has a default list that you can choose to disable to help keep the word cloud less cluttered. You also can control which words are excluded by providing your own words to remove in the Formatting area of the Visualization pane with this visual. I could add multiple words to the list as well, by using a comma. In this way, I can remove clutter.

In a sentiment analysis, certain words might have "weights." You can add these weights by using the values portion of the Visualization pane. From the download sample, you can see an example word cloud with default exclusions in Figure B-10.

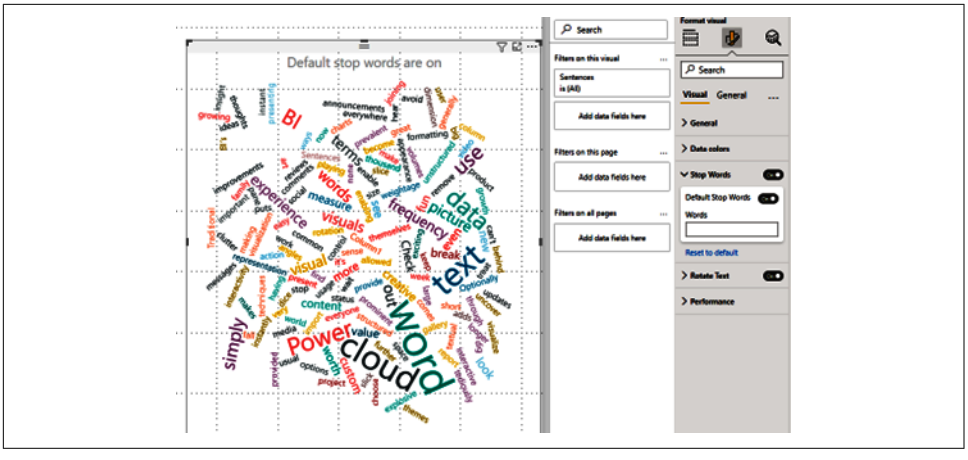


Figure B-10. Make judicious use of the Stop Words feature

Charticulator

Finally, we have *Charticulator*, one of the most exciting things to happen to custom visuals. Charticulator allows you to make custom visuals of your own without code. You can then import that visual you make at the Charticulator website into your Power BI report, and Charticulator is also completely certified for Power BI service.

Microsoft makes several video tutorials available, creating example visuals at Charticulator Video Tutorials, and that is worth reviewing if you want to try creating your own visuals.

Generally, if I can choose between creating a custom visual from scratch in Charticulator versus doing a little extra work to get a visual to do what I want in Power BI Desktop, I'll choose the latter. However, if I have a unique client request, I will sometimes find myself in Charticulator to build that perfect piece of the puzzle.

Be aware that Charticulator does have a learning curve. However, it also has the advantage of not being a code-based solution, which would prevent, say, an R- or Python-based visual from needing a personal data gateway.

Charticulator is Microsoft's secret weapon in the Power BI ecosystem that not enough people seem to know about or use. As with any product, I suggest just getting your hands on it and trying it out. It does not require any licensing, and you can add your own data to work from. Charticulator might be that final tool in your tool belt, but it is an incredibly powerful tool.

Index

C

D

E

F

G

H

I

O

P

Q

R

S

T

U

V

W

X

About the Author

Jeremey Arnold is the senior analytics architect at Onebridge, a large data analytics consulting firm in Indianapolis, Indiana. Jeremey has worked in data analytics for over a decade and has been a Microsoft Power BI user since its release in 2013. His experience covers multiple industries including healthcare, finance, manufacturing, and the public sector, all with a focus on transforming data into insights and enabling truly data-driven environments.

Colophon

The animal on the cover of *Learning Microsoft Power BI* is an East African hippopotamus (*Hippopotamus amphibius kiboko*), an extinct subspecies of the common hippopotamus (*Hippopotamus amphibius*). Common hippos, also known as large hippos, currently inhabit the waters of sub-Saharan Africa's rivers, swamps, and lakes.

The name "hippopotamus" is derived from the ancient Greek for "river horse." Hippopotamuses have a bulky, gray-brown barrel-shaped body carried on squat legs and a head of considerable size (the mouth is typically over a foot wide and can hinge open 150°). Hippos are the third-largest living land mammal—after elephants and white rhinoceroses—with males weighing 3.5 tons on average.

Many of the hippo's physical features relate to its amphibious nature, allowing it to traverse land and water surprisingly swiftly and with ease. Their feet have webbed toes that distribute weight evenly and provide powerful propulsion through the water. A hippopotamus's weight allows it to walk underwater, and its ears and nostrils can seal shut, allowing it to remain completely submerged for about five minutes.

Originally thought to be closely related to pigs and similar even-hooved mammals, DNA and fossil records have since provided evidence that hippo's closest relatives are sea mammals: dolphins, porpoises, and whales.

While the hippopotamus subsists on nighttime grazing on riverbanks and surrounding grassland, it is considered to be territorially aggressive and has frequently been reported charging and attacking humans (among other animals), sometimes resulting in fatalities.

The cover illustration is by Karen Montgomery, based on an antique line engraving from Cassell's *Popular Natural History.*. The cover fonts are Gilroy Semibold and Guardian Sans. The text font is Adobe Minion Pro; the heading font is Adobe Myriad Condensed; and the code font is Dalton Maag's Ubuntu Mono.