



University
of Glasgow

Excel - Introduction to Power Queries

V1.0)

e-mail: training@glasgow.ac.uk

web: gla.ac.uk/services/it/training

copyright © University of Glasgow

Course content created by Blair Thompson

Last edited by Blair ThompsonBlair Thompson on 01/06/26

Contents

Introduction	iii
Objectives	iii
Excel - Introduction to Power Queries	4
1 Introduction — What is Power Query?.....	4
2 The Power Query Editor	6
3 Getting Data into Power Query	13
4 Shaping Data — Columns and Rows	20
5 Shaping Data — Text Values	26
6 Filtering and Sorting	33
7 Adding a Calculated Column	37
8 Appending Tables	42
9 Merging Tables	47
Useful Shortcut keys	52

Introduction

Modern Excel is often used to bring together data from several different sources — spreadsheets exported from a reporting system, CSV files downloaded from a website, tables pasted from email, or a folder of monthly files that all share the same layout. Before that data can be analysed, it usually needs to be cleaned and reshaped; and when the source is updated, the same cleaning work tends to have to be repeated.

Power Query is the Excel feature built to solve exactly this problem. It lets you connect to a data source, record the steps needed to tidy and reshape the data, and then re-run those steps with a single click the next time the source changes. Power Query can also combine data from more than one table, either by stacking records from similar sources or by looking up related information across two tables.

This course introduces the Power Query Editor and walks through the transformations that participants are most likely to use day-to-day. No prior experience of Power Query, databases, or query languages is assumed.

Objectives

- On successful completion of this course participants will be able to:
- Describe what Power Query is and the problems it solves
- Navigate the Power Query Editor and identify its main components
- Import data from Excel tables, other workbooks, CSV files and folders
- Promote headers and assign correct data types
- Clean and reshape data using common Power Query transformations
- Append two or more tables to combine records from similar data sets
- Merge tables to bring related information together using a lookup column
- Control how a query loads into the workbook and refresh the data on demand

Excel - Introduction to Power Queries

1 Introduction — What is Power Query?

a. What is Power Query?

Power Query is the tool built into Excel for connecting to data, cleaning and reshaping it, and then refreshing the result on demand. It sits on the Data ribbon and has been part of Excel since 2016; in current versions of Excel it is found under the heading Get & Transform Data.

At its simplest, Power Query is a recorder. Every click you make while tidying a dataset — removing a column, changing a data type, filtering out blanks — is written down as a step. Those steps become a query, and the query can be re-run against the same source, or against a fresh version of it, as many times as you like.

The value of recording the work is straightforward: once a dataset has been cleaned properly, you never have to clean it the same way again. The query does it for you.

b. The problems Power Query solves

Many Excel users spend a surprising amount of time on the same small tasks, every week or every month:

Downloading a report from a system and tidying it up by hand before it can be used.

Copying and pasting a folder of monthly files into one combined sheet.

Re-building a report whenever the source data is updated.

Bringing together information from two tables that share a common key, such as a Department ID or a Customer Number.

Each of these tasks is manageable on its own, but they add up. Multiplied across a team, a semester, or a research project, they consume a serious amount of time — and every manual step is a chance to introduce an error.

Power Query is designed specifically to solve this class of problem. You do the cleaning once, in a way Excel can remember, and a single Refresh button repeats the whole process on the latest data.

c. The Connect – Transform – Load – Refresh cycle

It is helpful to think of Power Query as a four-step cycle. Every query you build follows the same pattern:

Connect. Point Excel at a source — an existing Excel table, another workbook on disk, a CSV file, or a folder containing several files in the same layout. Power Query creates a live connection to the source; it does not copy the data into your workbook.

Transform. Open the Power Query Editor and make the changes you would normally make by hand: promote the first row to headers, set sensible data types, remove blank rows, split a column, rename a column, filter out records you do not want. Each change is recorded as a step in the Applied Steps pane on the right.

Load. When you are happy with the shape of the data, choose how Excel should deliver the result: as a new worksheet table, as a connection only, or into the Data Model. Most of the time, a simple table on a worksheet is exactly what you want.

Refresh. Later, when the source changes, click Refresh. Power Query re-runs the recorded steps against the new version of the source and updates the table in Excel. There is no re-cleaning to do.

These four stages appear over and over again throughout the rest of the course. The remaining chapters each focus on one or two of them in detail.

d. Where to find Power Query in Excel

Everything you need sits in one place on the Data ribbon, in the Get & Transform Data group. The four options below cover the data sources that come up most often:

From Table/Range. Use this when the data is already inside the current workbook and is formatted as an Excel Table (or you are happy for Excel to convert a range into a Table first). This is the quickest way to start experimenting with Power Query, because the source is right in front of you.

From Text/CSV. Opens a .csv or .txt file on disk and starts a new query from it. Power Query previews the file and lets you confirm the delimiter and encoding before you begin transforming.

From Workbook. Opens a different Excel file on disk. Useful when the data you need lives in a separate workbook you do not want to copy into your own file.

From Folder. Points Power Query at every file in a folder and combines them into one query. This is the most powerful of the four and is the usual answer to “I have a folder of monthly files to stitch together.”

Finished queries — the ones you have created and saved — appear in the Queries & Connections pane on the right of the Excel window. If the pane is hidden, turn it on with View ribbon → Queries & Connections, or press CTRL + ALT + L.

e. When should I reach for Power Query?

Power Query is worth the effort of learning in any of the following situations:

You find yourself doing the same clean-up on the same type of file, week after week.

You need to combine several files or sheets that share the same structure.

Your source data is almost — but not quite — in the shape you need.

A report should stay in sync with a changing data source, rather than being a fixed snapshot.

A colleague will need to re-run your analysis on next month's data with a single click.

If none of these sounds familiar, a traditional Excel approach with formulas, sort and filter may well be enough. Power Query is not a replacement for everyday Excel — it is a companion to it, aimed at the jobs that are too repetitive, too error-prone, or too structural to solve with cell formulas alone.

f. What Power Query is not

A few quick myth-busters, because Power Query can look daunting at first glance:

Not a replacement for Pivot Tables. Power Query prepares the data. Pivot Tables summarise it. The two are complementary: a well-built Power Query feeds a clean table into a Pivot Table, and the Pivot Table then answers the analytical questions.

Not a programming language. Under the hood, every query is written in a language called M — but in practice you very rarely need to read it, and you almost never need to write it. The whole Power Query Editor is designed to let you build M code without ever seeing it.

Not Power BI. Power BI uses the same Power Query engine, so many of the skills you learn in this course transfer directly. The difference is simply where the result is delivered: Power Query in Excel feeds an Excel workbook; Power Query in Power BI feeds a Power BI report.

Not just for experts. If you are comfortable using AutoFilter, copying a column, and working with Excel Tables, you have enough Excel to use Power Query productively. Most of the work is done with the mouse, on familiar-looking menus.

g. A note on Excel versions

This course is written against Microsoft 365 and Excel 2021 on Windows. Power Query is also available in earlier versions of Excel (2016 and 2019), and in Excel for Mac, although the layout and naming of a few menus differ.

Wherever a menu item or ribbon name is mentioned, the manual uses the current Microsoft 365 wording. If you are following along on an older version of Excel and cannot find something, please ask — the same feature will almost certainly be there, and often in a very similar place.

Next: Chapter 2 introduces the Power Query Editor itself — the window you land in after clicking one of the Get Data options above.

2 The Power Query Editor

Chapter 1 explained what Power Query is and the four-stage cycle it follows: Connect, Transform, Load, Refresh. This chapter is a guided tour of the window where the Transform stage happens — the Power Query Editor. You will not build a finished query yet; the aim is simply to get comfortable with the layout, the names of the panes, and the two options for leaving the Editor when you are finished.

The practice file for this chapter is **2 - The Power Query Editor.xlsx**. It contains a single Excel Table called Bookings with a little over five thousand rows of registration data from last summer's Student Research Symposium. Every chapter has its own fresh copy of the data — if something goes wrong, you can simply close this file without saving and open the Chapter 3 copy when you get there.

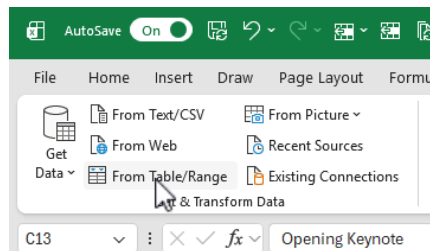
a. Getting to the Editor

The Power Query Editor is not opened from the Start menu — it is launched from inside Excel, from any of the Get Data options on the Data ribbon. Pointing Excel at a source is what triggers the Editor to appear.

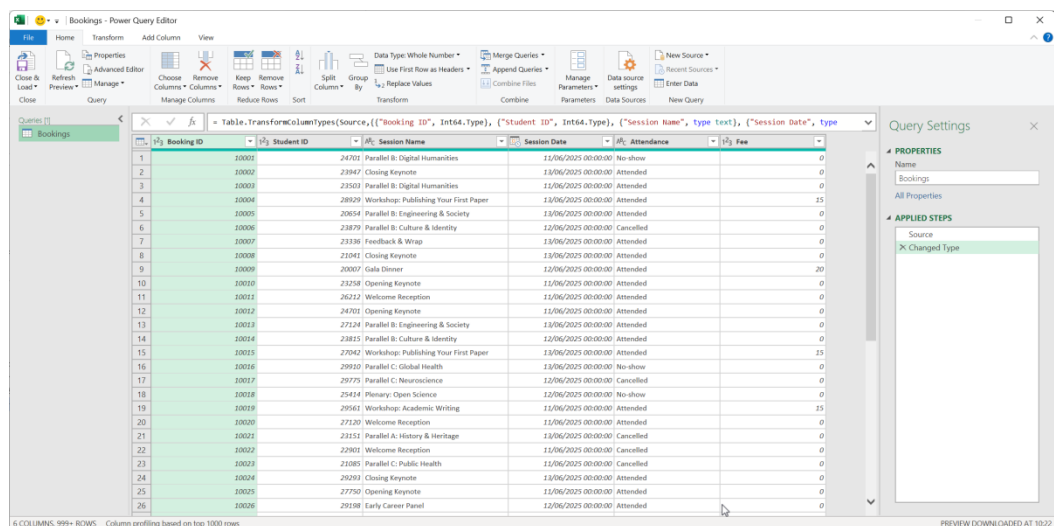
The quickest way in, and the one used throughout this course, is From Table/Range. This takes an Excel Table that is already in the current workbook and opens it as a new query.

Exercise: Open the Bookings table in the Power Query Editor

- 1 Open **2 - The Power Query Editor.xlsx** from your Practice Files folder.
- 2 Click anywhere inside the Bookings table.
- 3 On the Data ribbon, in the Get & Transform Data group, click From Table/Range.



- 4 Wait for the Power Query Editor to open in its own window. Leave it open for the rest of the chapter.

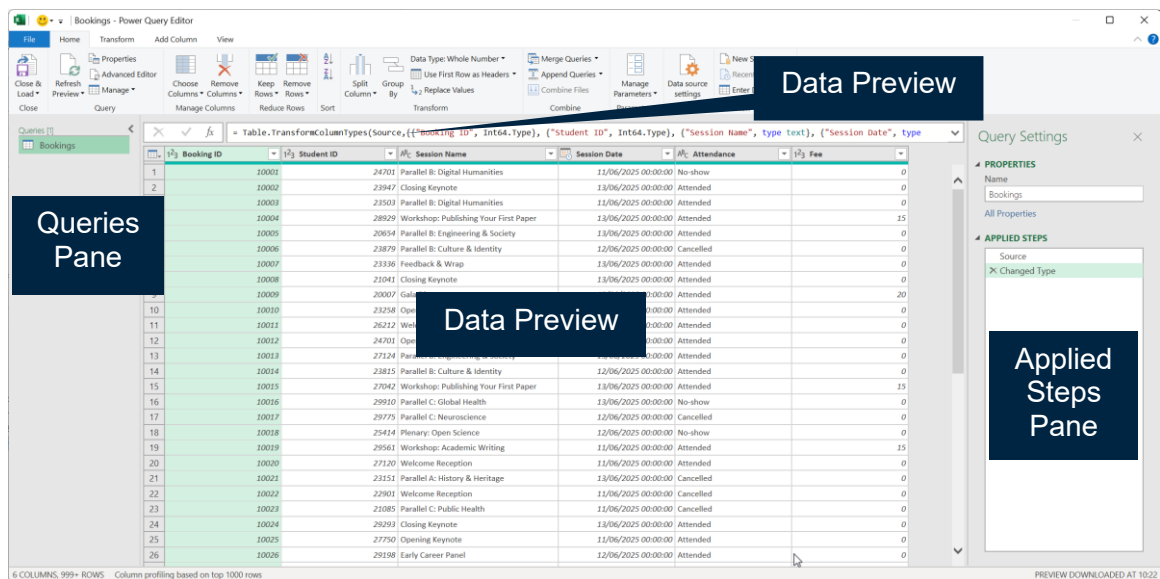


If the Editor opens behind the Excel window, use ALT + TAB to bring it to the front. This happens often enough that it is worth knowing about from the start.

b. The Editor window at a glance

The Editor is a separate window from Excel. It has its own title bar, its own ribbon, and its own close button. Crucially, closing the Editor does not close Excel — and closing Excel while the Editor is still open will usually leave the Editor running behind the scenes.

The window is divided into four main regions:



Queries pane. The narrow strip on the left. It lists every query in the current workbook.

Data Preview. The large table in the middle. This is a sample of the data as it stands after the current step.

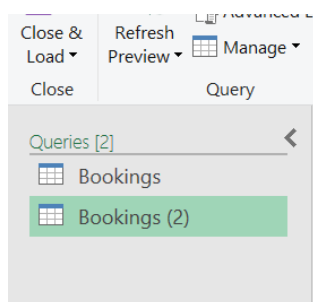
Applied Steps pane. The panel on the right. Each step you apply is written here, in order.

Formula Bar. The single line just above the Data Preview, showing the M code for the selected step.

Above all of this sits the Editor ribbon, with its own Home, Transform, Add Column and View tabs. Below everything is a status bar that shows how many columns and rows are currently visible in the preview.

c. The Queries pane

The Queries pane is a table of contents for the workbook's queries. A workbook can hold as many queries as you like, and each one appears here by name.



Clicking a query in this pane is how you move between queries while the Editor is open. The Data Preview, Applied Steps and Formula Bar all update to show whichever query is currently selected.

For now, the pane should contain a single entry named Bookings — the query Excel created the moment you clicked From Table/Range.

Later in the course, when a single workbook contains half a dozen queries at once, this pane becomes the main way of finding your place.

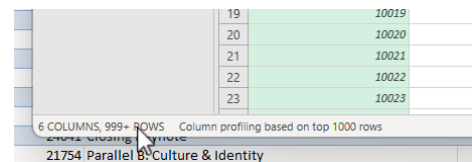
A query's name in this pane is also the name used when the query is loaded back into Excel. Renaming a query here is the cleanest way to rename the resulting table.

d. The Data Preview (it is only a sample)

The large grid in the middle of the Editor shows the data as it currently stands. As you apply steps, this preview updates to reflect the effect of each one. It looks like an ordinary Excel sheet, but there are two important differences.

It is a preview, not the data. No changes are being made to the source workbook, or to any sheet in Excel. Everything you do here is recorded as instructions; Excel only runs those instructions when you ask it to load or refresh the query.

It is a sample, not the full dataset. To keep the Editor responsive, Power Query only loads the first thousand rows or so into the preview. The Bookings table has around five thousand rows, but the preview will typically show a subset of them. The status bar at the bottom of the window confirms this — it shows the number of columns and rows currently visible, which is usually less than the total.



19	10019
20	10020
21	10021
22	10022
23	10023

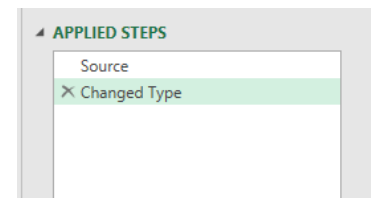
6 COLUMNS, 999+ ROWS Column profiling based on top 1000 rows
21754, Parallel 8, Culture & Identity

This distinction matters. A common source of confusion is to filter a column in the preview, see that it “worked” on a few hundred rows, and assume the filter is complete. In reality, the step has been recorded against the query and will be applied to every row when the query is loaded, including rows you never saw in the preview. The preview is there to design against, not to verify against.

e. The Applied Steps pane

Every transformation you apply is recorded as a named step in the Applied Steps pane on the right. The list reads top-to-bottom in the order the steps were applied.

At this point, even without doing any work, the pane will usually already show one or two automatic steps:



Source. The first step in every query. Records where the data came from.

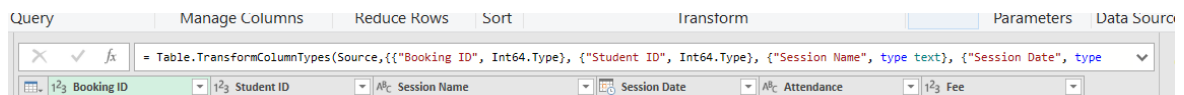
Changed Type. Power Query's best guess at the data type of each column — added automatically as soon as a table is loaded. You can keep it, alter it, or delete it.

Clicking a step moves the Data Preview back to the state the data was in immediately after that step ran. This makes the Applied Steps pane the single most useful troubleshooting tool in Power Query: if a query goes wrong, you can click backwards through the steps until you find the point at which the result stops looking right.

Steps can be renamed, reordered, or deleted. Renaming in particular is worth the small effort — a step called Removed Columns1 is much less helpful a month later than one called Removed Personal Details.

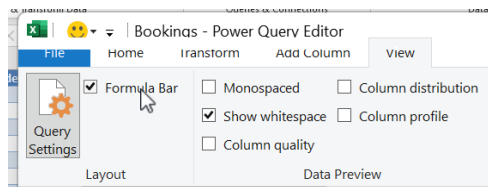
f. The Formula Bar

Immediately above the Data Preview is a single line that shows the M code for whichever step is currently selected. M is the language Power Query uses under the hood.



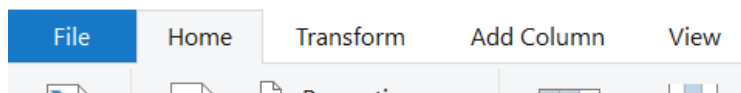
In an introductory course, you will very rarely need to edit this directly. It is shown in this chapter only so that you recognise it when you see it, and understand that every click in the Editor is being translated into one line of M underneath. Later chapters point out a few small, safe edits the Formula Bar is useful for.

If the Formula Bar is not visible, turn it on via View ribbon → Layout → Formula Bar.



g. The Editor ribbon

The ribbon at the top of the Editor groups the available transformations into four tabs:



Home. The most-used transformations — remove columns, remove rows, filter, sort, change type, Close & Load. Most of the work in a typical query is done from here.

Transform. Operations that change an existing column in place: splitting, replacing values, trimming, changing case, rounding numbers, extracting parts of a date.

Add Column. Operations that create a new column from one or more existing ones, leaving the originals alone. Useful when you want to derive a value without losing the original.

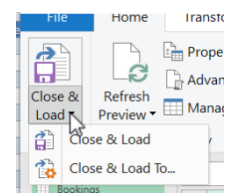
View. Turns window features on and off — Formula Bar, Column Quality, Column Distribution, and the Query Dependencies diagram. The defaults are sensible; you rarely need to change them.

Every button on these tabs produces a step in the Applied Steps pane. Nothing is ever done silently.

h. Leaving the Editor — Close & Load vs Close & Load To

When you are finished with a query, the Home ribbon offers two ways to leave the Editor. They look similar but behave very differently, and the choice you make determines where the result ends up.

Close & Load. Closes the Editor and loads the result straight onto a new worksheet in Excel as a Table. This is the right choice when you want to see the finished data in the workbook.



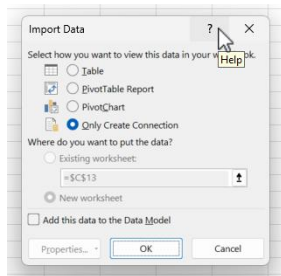
Close & Load To... Closes the Editor and opens a small dialog asking where the result should go. The options are: as a Table on a worksheet (same as above), as a Connection Only (no sheet at all — useful when the query is a helper that other queries will build on), or into the Data Model.

For everyday use, Close & Load is perfectly fine. Connection Only becomes important later when you start combining several queries into one final output — you usually do not want every intermediate query to produce its own sheet.

Task: Close the Editor as a Connection Only

Still inside the Power Query Editor, click the Home ribbon.

- 1 Click the small arrow beneath Close & Load and choose Close & Load To...
- 2 In the Import Data dialog, select Only Create Connection, then click OK.



- 3 Back in Excel, open the Queries & Connections pane (View ribbon, or CTRL + ALT + L) and confirm that the Bookings query now shows “Connection only” beside it.

No new sheet has been created in the workbook. The query is loaded and ready to use, but the data has not been written anywhere visible. This is exactly what Connection Only means.

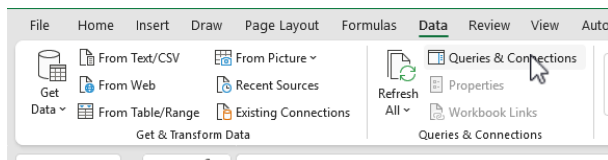
i. Re-opening a saved query

A query is saved with the workbook it belongs to. Re-opening a query simply means re-opening the workbook and double-clicking the query’s name in the Queries & Connections pane. The Editor will appear with every step exactly as it was left.

Task: Re-open the Bookings query

Make sure you are in Excel (not the Editor).

- 1 If the Queries & Connections pane is not already visible, press **Data – Queries and Connections – Queries and Connections** (or CTRL + ALT + L) to open it.



- 2 Double-click the Bookings query. The Power Query Editor opens with the Applied Steps pane showing exactly the steps that were in place when the query was last closed.

The screenshot shows the Microsoft Excel Power Query Editor interface. The main area displays a table named 'Bookings' with the following data:

Booking ID	Student ID	Session Name	Session Date	Attendance
10001	24701	Parallel & Digital Humanities	11/06/2025 00:00:00	No-show
10002	23947	Closing Keynote	13/06/2025 00:00:00	Attended
10003	23503	Parallel & Digital Humanities	11/06/2025 00:00:00	Attended
10004	28929	Workshop: Publishing Your First Paper	13/06/2025 00:00:00	Attended
10005	20654	Parallel & Engineering & Society	13/06/2025 00:00:00	Attended
10006	23879	Parallel & Culture & Identity	13/06/2025 00:00:00	Cancelled
10007	23336	Feedback & Wrap	13/06/2025 00:00:00	Attended
10008	21041	Closing Keynote	13/06/2025 00:00:00	Attended
10009	20007	Gala Dinner	12/06/2025 00:00:00	Attended

On the right side, the 'Queries & Connections' pane shows a list of queries, including 'Bookings' and 'Connections'. The 'Bookings' query is selected.

3 Close the Editor and close Excel without saving

Being able to leave a query, close Excel for the day, and pick up exactly where you left off is one of the quieter strengths of Power Query. The query is part of the workbook, so wherever the workbook goes, the query goes with it.

Next: Chapter 3 starts the Transform stage properly, using From Table/Range to build the first real query against the Bookings table.

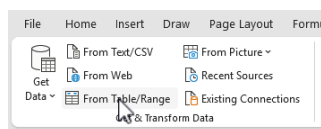
3 Getting Data into Power Query

Chapter 2 showed you around the Power Query Editor. This chapter is about how to get to it — or more precisely, about the four most common ways of pointing Power Query at a source of data. By the end of the chapter you will have built four separate queries, one from each source, and seen how little the Editor changes regardless of where the data originally came from.

a. Overview of the four sources

All four sources sit on the Data ribbon, inside the Get & Transform Data group. They are presented as a short list of buttons; choosing one of them starts a new query and opens the Editor.

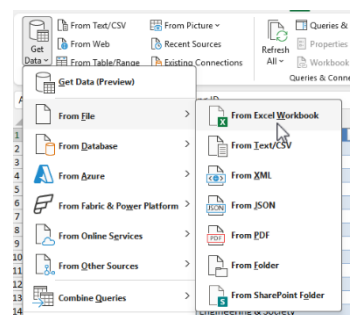
From Table/Range. The data is already inside the current workbook, formatted as an Excel Table. This is the fastest way to experiment.



From Excel Workbook. The data lives in another .xlsx file on disk. Power Query will list the sheets and tables inside that file so you can pick one.

From Text/CSV. A plain-text file with delimited values — typically a .csv exported from another system. Power Query shows a preview and confirms the delimiter before loading.

From Folder. A whole folder of files that share the same structure (for example, a monthly file per department). Power Query combines all of them into one query.



b. Practice files for this chapter

Chapter 3 uses four separate practice files, one for each source. They all live in your Practice Files folder:

- 3a - Getting Data - Bookings.xlsx — the main conference bookings workbook; used for the From Table/Range task.
- 3b - Getting Data - Student List.xlsx — a separate workbook of student records; used for the From Workbook task.
- 3c - Getting Data - Bookings.csv — a CSV export from the conference registration system; used for the From Text/CSV task.
- 3d - Getting Data - Daily Bookings — a folder containing three daily booking files; used for the From Folder task.

You will not need to save any changes in this chapter — every task ends with a Connection Only load. If anything goes wrong, close the workbook without saving and start the task again from the top.

c. From Table/Range

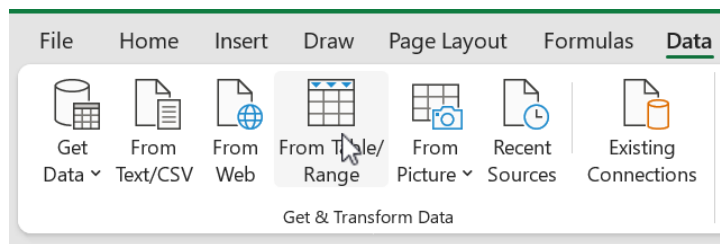
From Table/Range starts a query from an Excel Table in the current workbook. The only prerequisite is that the data is formatted as a Table — if it is still a plain range when you

click From Table/Range, Excel will offer to convert it for you, a conversion you should almost always accept.

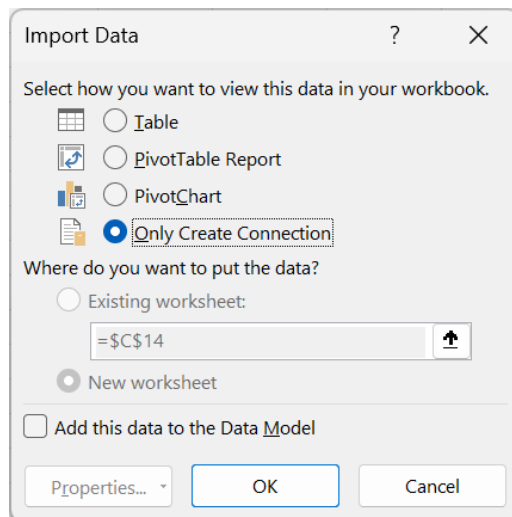
You used this source in passing in Chapter 2. Here the task is framed more deliberately, as part of a set.

Exercise: Load Bookings using From Table/Range

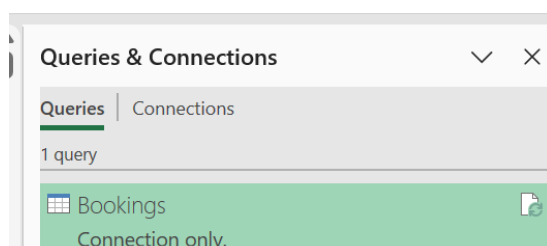
- 1 Open 3a - Getting Data - Bookings.xlsx from your Practice Files folder.
- 2 Click anywhere inside the Bookings table on the Bookings sheet.
- 3 On the Data ribbon, in the Get & Transform Data group, click From Table/Range.



- 4 The **Power Query Editor** opens with a new query named **Bookings**. Do not change anything yet.
- 5 On the Home ribbon, click **Close & Load to**
- 6 In the **Import Data** dialog, select **Only Create Connection**, then click **OK**.



- 7 Back in Excel, open the Queries & Connections pane (CTRL + ALT + L) and confirm that Bookings is listed



Leave this workbook open. The remaining tasks in this chapter all happen in the same workbook — you are adding three more queries alongside Bookings, not starting a new file each time.

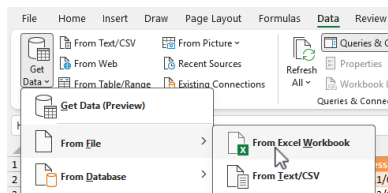
d. From Workbook

From Workbook is used when the data you want lives in a different Excel file. Pointing Power Query at that file does not copy it into your workbook — instead, Power Query creates a connection to the other file, and reads from it every time the query is refreshed.

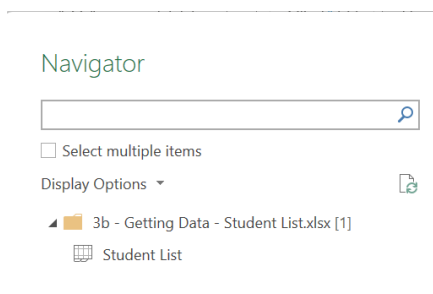
The first step is a file-browser dialog. Once you have picked the file, Power Query shows a Navigator window listing every sheet, range and Table inside it. You pick the one you want and click OK; Power Query then opens the Editor just as it did for From Table/Range.

Exercise: Load Student List using From Workbook

- 1 With **3a - Getting Data - Bookings.xlsx** still the active workbook, go to the Data ribbon.
- 2 In the **Get & Transform Data** group, click **Get Data** → **From File** → **From Excel Workbook**.

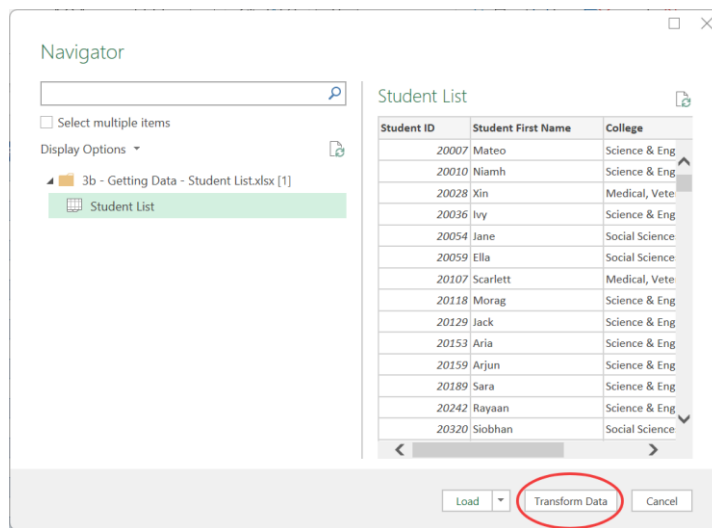


- 3 In the file browser, navigate to your Practice Files folder and choose **3b - Getting Data - Student List.xlsx**, then click **Import**.
- 4 When the Navigator window appears, select the **Student List** sheet on the left. A preview of the student records will appear on the right. Click **Transform Data**.

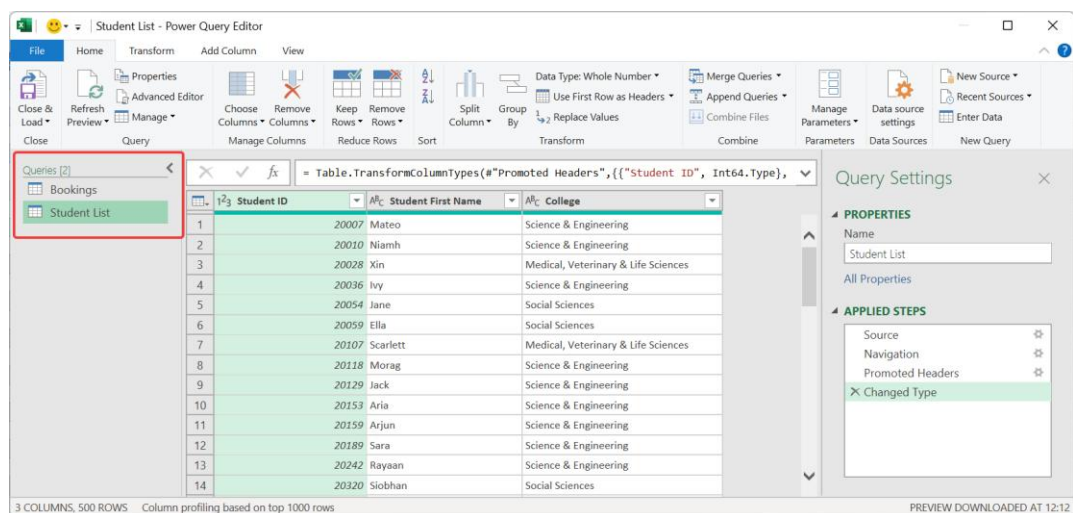


- 5 The **Power Query Editor** opens with a new query named **Student List**. Do not change anything.

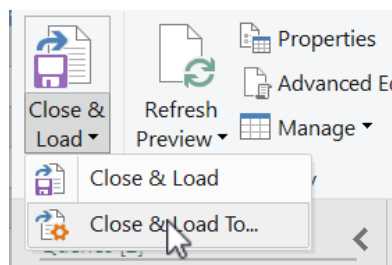
6 Click **Close & Load**



7 In the **Queries & Connections** pane you should now see two queries: **Bookings** and **Student List**



8 Select **Close and Load** and then **Close & Load To**



9 From the Import Data dialog box, Select **Only Create Connection**

10 Click **OK**

Note - Power Query noted the full file path of 3b when you imported it. If you later rename or move that file, the query will fail to refresh — the query is pointing at a location on disk, not at the file itself.

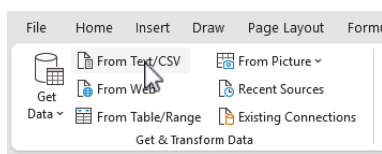
e. From Text/CSV

A CSV file is a plain text file containing a row per record, with values separated by commas (or occasionally semicolons or tab characters). CSV is the most common way of moving data out of other systems — a student records system, a booking system, a survey tool — because every piece of software can read and write it.

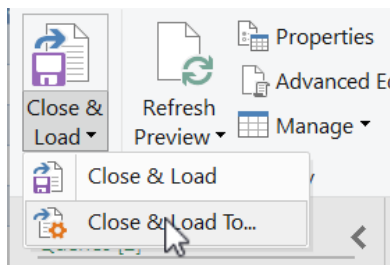
From Text/CSV shows a small preview dialog before the Editor opens. In this dialog Power Query guesses the delimiter, the file encoding, and how the data types in each column should be interpreted. You should glance at the preview and confirm that the columns look right before clicking Transform Data.

Exercise: Load the CSV follow-up cohort using From Text/CSV

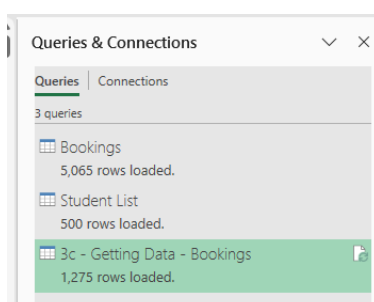
- 1 Still in the same workbook, go to the Data ribbon and click From Text/CSV.



- 2 Browse to your Practice Files folder and choose **3c - Getting Data - Bookings.csv**, then click **Import**.
- 3 In the preview dialog, take a moment to look at the preview. You may notice the first row is not a header row — that is a deliberate quirk of this export, and you will clean it up in a later chapter. Do not change anything in the dialog for now.
- 4 Click **Transform Data**. The Power Query Editor opens.
- 5 Click **Close & Load** and then **Close and Load to**



- 6 Click **Only Create Connection**
- 7 Click **OK**
- 8 The Queries & Connections pane now shows three queries:



f. From Folder

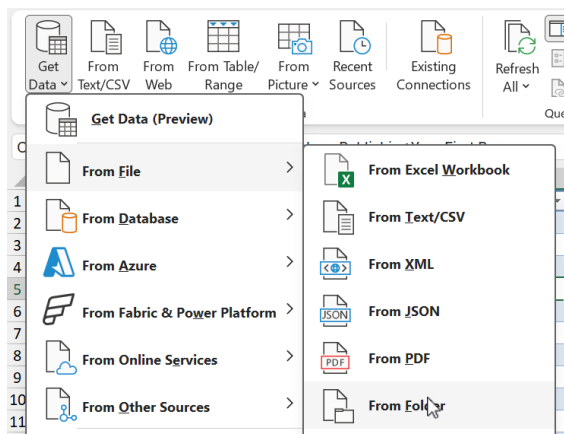
From Folder is the most powerful of the four sources, and the one that changes how people feel about Power Query. You point it at a folder on disk, and it combines every file in that folder into a single query — assuming the files share the same structure.

This is the answer to “I have a folder of monthly files.” It is also the answer to “Every department sent me a spreadsheet with the same layout.” Rather than opening each file and copy-pasting, you let Power Query walk the folder for you.

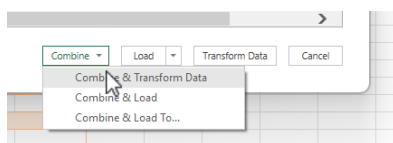
The folder 3d - Getting Data - Daily Bookings contains three files — one per day of the symposium. Each file has the same columns as the main Bookings table, but covers a single day’s worth of bookings. Combined, they contain everything in the Bookings table (with one or two small differences you will come across later).

Task: Combine the daily files using From Folder

- 1 On the Data ribbon, click **Get Data** → **From File** → **From Folder**.

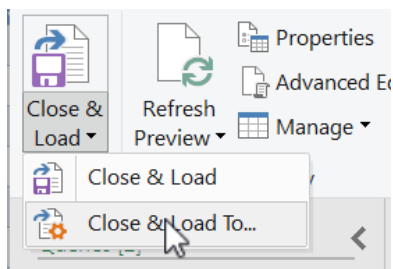


- 2 Browse to your Practice Files folder and select the **3d - Getting Data - Daily Bookings** folder, then click **Open**.
- 3 Power Query shows a list of the files it has found in the folder — three files in this case. Click the small arrow beside **Combine** at the bottom of the dialog and choose **Combine & Transform Data**.



- 4 A second dialog appears asking which sheet from a sample file to use as the template. Select **Day 1** (or the only sheet listed), then click **OK**.
- 5 The Editor opens with a new query called **3d - Getting Data - Daily Bookings**. The Applied Steps pane will show several automatic steps — Power Query has done quite a lot of work for you.

- 6 Click the **Close & Load** and then **Close & Load to**



- 7 Select **Only Create Connection**

- 8 Click **OK**

The Queries & Connections pane now shows four connection-only queries, plus a small number of helper queries that From Folder added automatically. Those helpers are normal — leave them alone.

From Folder adds a column called `Source.Name` to the combined result, showing which file each row came from. This is often more useful than the row data itself — it lets you trace a record back to the specific monthly (or daily) file that contained it.

g. Where we are now

You have built four queries from four different sources and loaded all of them as Connection Only. Nothing is written to any sheet; no transformations have been applied; the workbook is, for all practical purposes, still empty. And yet the Queries & Connections pane is now carrying a live connection to a workbook table, another workbook, a CSV file and a folder of files all at once.

This is the point of the Connect stage. From the next chapter onwards, the focus shifts to the Transform stage — cleaning and reshaping the data inside these queries — and the loading options (Connection Only versus Table) will start to matter for a different reason.

Next: Chapter 4 starts the Transform stage by shaping data at the column and row level — removing what you do not need, promoting headers, and setting sensible data types.

4 Shaping Data — Columns and Rows

Chapter 3 was all about connecting to data. This chapter is where the Transform stage of Power Query begins in earnest. You will open a deliberately untidy workbook and, one step at a time, turn it into a clean table — without touching the source file.

The transformations in this chapter operate on the structure of the table: which rows are in it, which columns are in it, what the headers are, and what the data types are. Chapter 5 will then tackle the contents of individual cells.

a. The practice file for this chapter

This chapter uses a single practice file:

4 - Shaping Data - Columns and Rows.xlsx — an Excel workbook that looks like a real-world export from a registration system. It contains a Table called Export on a sheet called Export.

Open the file and have a look at the Export sheet before you start the tasks. You should see:

A row of auto-generated column headers (Column1, Column2, ...) at the top.

A metadata line on the first data row — something like “CONFERENCE REGISTRATIONS EXPORT — Generated ...” in column 1.

A blank row below that.

The real column headers — BOOKING_ID, STUDENT_ID, SESSION_NAME, SESSION_DATE, ATTENDANCE, FEE, NOTES — sitting on the third data row.

A few hundred booking records below the real headers, with the occasional blank row scattered through.

An untidy footer row at the bottom — something like “=== End of export ... ===”.

A NOTES column that is almost entirely empty, with a handful of cells holding things like “Gluten-free lunch” or “Wheelchair access”.

This is a deliberately messy file. Real registration-system exports very often look exactly like this. Everything you learn in this chapter is aimed at tidying such files up in Power Query rather than by hand.

b. Every click becomes a step

Before you touch any buttons, there is one habit to build from the start of this chapter: watch the Applied Steps pane on the right. Every transformation in this chapter adds a line to it.

This matters because the order of the steps is the order in which they will run when the query is refreshed. Removing top rows before promoting headers is different from promoting headers before removing top rows. Power Query is patient about letting you experiment — you can delete or reorder steps later — but it is much less work to apply them in the right order the first time.

For this chapter, the order below is the recommended order. Later chapters will show examples where a slightly different order is necessary.

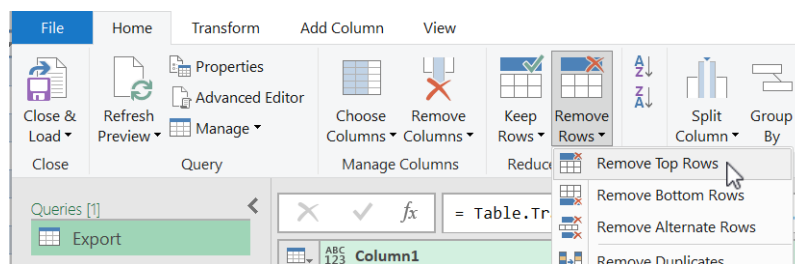
c. Remove Top Rows

The first job is to get rid of the junk at the top of the file: the metadata row and the blank row below it. Both are inside the Table, so Power Query treats them as data rows unless you tell it otherwise.

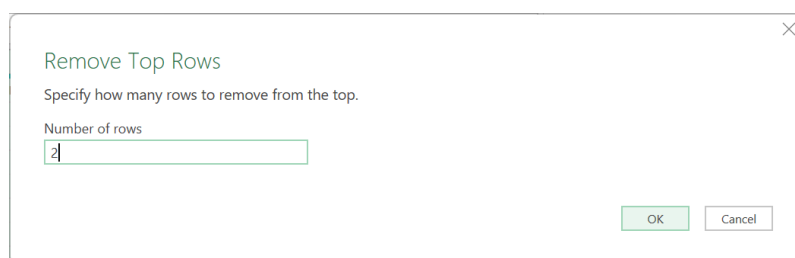
Remove Top Rows lives on the Home ribbon, inside the Remove Rows drop-down. It takes a single number — how many rows to discard from the top.

Task: Connect, then remove the junk at the top

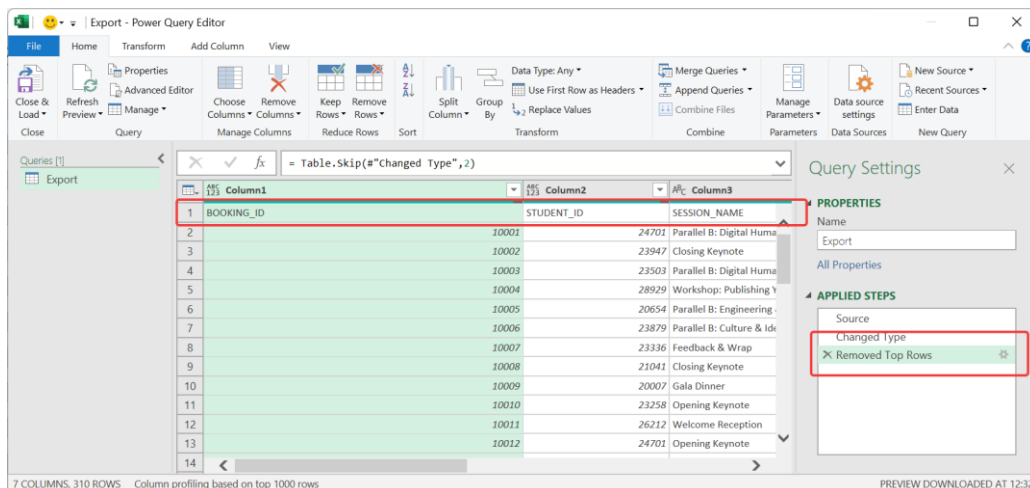
- 1 Open **4 - Shaping Data - Columns and Rows.xlsx** from your Practice Files folder.
- 2 Click anywhere inside the **Export** table on the **Export** sheet.
- 3 On the **Data** ribbon, click **From Table/Range**.
- 4 The Power Query Editor opens. The column headers are Column1, Column2, ... and the first row of data is the metadata line.
- 5 On the **Home** ribbon, click **Remove Rows** → **Remove Top Rows**.



- 6 In the small dialog, type 2 (the metadata row plus the blank row beneath it) and click OK.



- 7 The Data Preview should now start with a row of text beginning **BOOKING_ID**, **STUDENT_ID**, **SESSION_NAME** ... — the real headers.



- 8 Look at the Applied Steps pane on the right: a new step called Removed Top Rows has appeared.

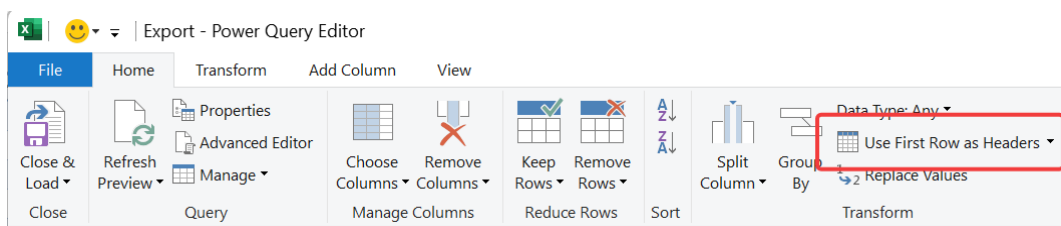
If the Editor did not open, or opened with clean-looking headers already in place, close it without loading, then open the Chapter 4 workbook again from a fresh copy. A previously-saved query in the same workbook can sometimes interfere.

d. Use First Row as Headers

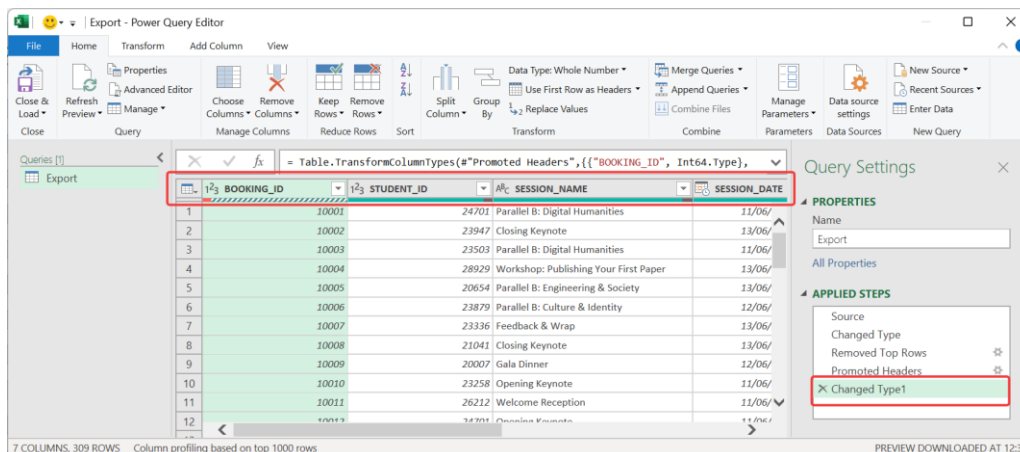
The real headers are now the top row of the preview, but Power Query still thinks the headers are Column1, Column2, ... (those came from the Excel Table itself). Promoting the top row will rename the columns properly.

Task: Promote the first row to headers

- 1 On the **Home** ribbon, click **Use First Row as Headers**.



- 2 The first row of data is pulled up into the header area. Your columns should now be **BOOKING_ID**, **STUDENT_ID**, **SESSION_NAME**, **SESSION_DATE**, **ATTENDANCE**, **FEE**, **NOTES**.



- Two new steps appear in the Applied Steps pane: **Promoted Headers** and, immediately afterwards, **Changed Type**. Power Query adds the second one automatically — that is the subject of the next section.

e. Change Type

The Changed Type step Power Query just added is its first guess at the data type of each column. Each column header now shows a small icon at the left — ABC for text, 123 for a whole number, a calendar for a date, and so on. Getting these right matters; many later transformations (filtering by date, grouping by number) will only work when the type is set correctly.

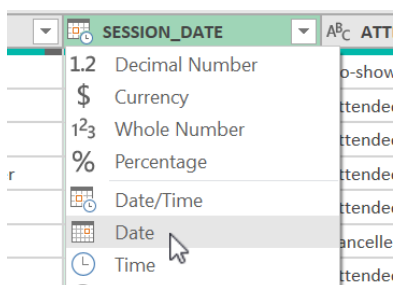
Power Query's guess is usually very close. Two pieces of advice are worth following:

Always check dates and numbers. Dates are the single biggest source of Power Query errors, particularly between UK and US date formats. Click the calendar icon at the left of **SESSION_DATE** and confirm it has been set to Date. If it has been set to Text, click the icon and change it to Date.

Leave text columns alone. Columns like **SESSION_NAME** and **ATTENDANCE** should stay as ABC (Text). There is rarely a reason to change them.

Task: Verify the data types

- Click the small icon at the left of the **SESSION_DATE** header. It should show a calendar icon. If it reads anything else, click it and choose Date from the list.



- Click the icon at the left of **FEE**. It should show an icon corresponding with **Whole Number** or **Decimal Number** — either is fine here.

- Click the icon at the left of **BOOKING_ID** and **STUDENT_ID**. Both should be Number.
- Leave ABC as the type for **SESSION_NAME**, **ATTENDANCE**, and **NOTES**.

If you need to change a type, do not edit the Changed Type step directly — click the column icon and Power Query will either update Changed Type, or add a new Changed Type 1 step. Either is fine for an introductory course.

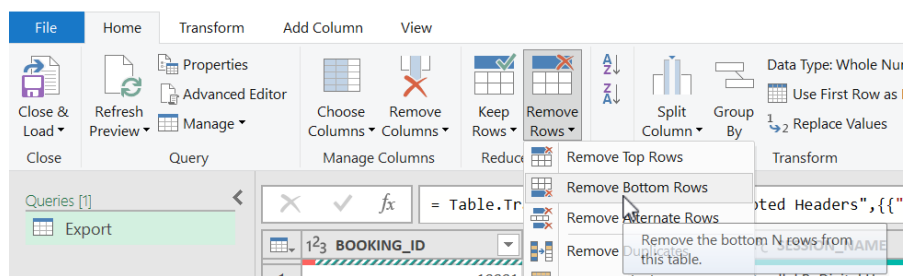
f. Remove Bottom Rows

The junk footer row at the end of the table (“=== End of export ... ===”) is still in the preview. You can confirm this by scrolling to the bottom, or by clicking the **SESSION_DATE** column header and noting that the footer row has a red error-like indicator because its text does not parse as a date.

Remove Bottom Rows is the counterpart to Remove Top Rows. It discards a specified number of rows from the bottom of the table.

Task: Remove the junk footer row

- On the **Home** ribbon, click **Remove Rows** → **Remove Bottom Rows**.



- In the dialog, type 1 and click OK.
- The “=== End of export ... ===” row is gone. A new step **Removed Bottom Rows** appears in **Applied Steps**.

g. Remove Blank Rows

Scattered through the data are a handful of completely blank rows — rows where every cell is null. These are common in real-world exports, especially from systems that group records by category and leave a blank row between groups.

Remove Blank Rows removes every row in which every single cell is empty. It leaves alone any row that has at least one value, so a row with a booking number but no notes is not affected.

Task: Remove blank rows

- On the **Home** ribbon, click **Remove Rows** → **Remove Blank Rows**.
- The blank rows in the preview disappear. A new step **Removed Blank Rows** appears in **Applied Steps**.

- 3 Glance at the status bar at the bottom of the window. The row count in the preview should drop by the number of blank rows that had been in the data — a small but visible change.

Remove Blank Rows is stricter than it looks. A row with a single space character in one cell is not a blank row from its point of view — the cell is not empty, it just looks empty. Chapter 5 shows how to deal with that case using **Trim**.

h. Remove Columns (and Choose Columns)

The **NOTES** column is almost entirely empty and is not needed for any of the analysis the course will go on to do. Removing it tidies the preview and speeds up every later step.

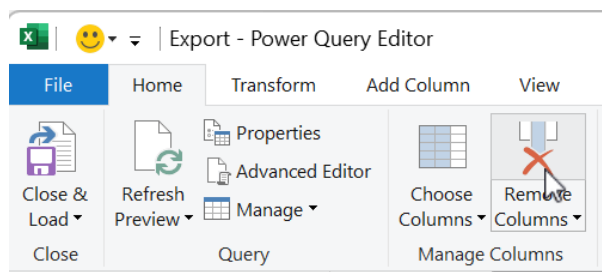
There are two ways to do this:

Remove Columns. Select the column you want to discard, then click Remove Columns on the Home ribbon. This is the best option when you want to get rid of one or two columns and keep the rest.

Choose Columns. Opens a dialog listing every column in the table. You tick the ones you want to keep. This is the best option when a wide table has many columns and you want to be explicit about which ones to retain.

Task: Remove the NOTES column

- 1 Click the **NOTES** column header to select the whole column.
- 2 On the **Home** ribbon, click **Remove Columns**.



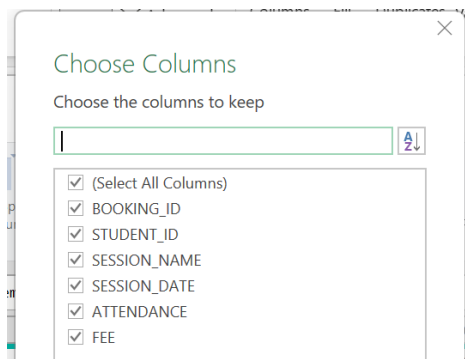
- 3 The **NOTES** column disappears. A new **Removed Columns** step is added to **Applied Steps**.

For contrast, briefly open Choose Columns to see what it looks like, but do not apply it:

Task: Look at Choose Columns (do not apply)

- 1 On the **Home** ribbon, click **Choose Columns**.

- 2 A dialog lists the six remaining columns with a tick-box beside each.



- 3 Cancel the dialog without changing anything — the point is just to see the tool.

Choose Columns and Remove Columns achieve the same kind of result. Pick whichever one is less work on the table in front of you. Both produce equally clear steps in the Applied Steps pane.

i. Loading the cleaned result

You have built a query that turns a messy export into a clean table of booking records. For this chapter, load the result back into Excel as a proper Table — not as a Connection Only — so you can see the effect of the work.

Task: Load the cleaned table into Excel

- 1 On the **Home** ribbon, click the top half of **Close & Load** (not the arrow).
- 2 The Editor closes and Excel creates a new worksheet containing the cleaned data as a Table.
- 3 Compare the new sheet to the original Export sheet. Same source, same file, very different shape.

The original Export sheet is untouched. All of the transformations you applied live only in the query, and Excel ran them one after another when you clicked Close & Load. If you later update the source sheet and click Refresh All on the Data ribbon, the query will re-run every step against the new data.

Next: Chapter 5 moves from the structural transformations of this chapter to transformations of the values inside individual cells — trimming whitespace, splitting text, changing case, and so on.

5 Shaping Data — Text Values

Chapter 4 reshaped the table — which rows and columns were in it, and what the headers and types were. This chapter zooms in one level: we will work on the contents of individual cells.

Almost every real-world dataset has these problems somewhere: trailing spaces in a name field that make two identical entries look different, tabs or line breaks smuggled in by a

copy-paste from a PDF, case inconsistencies that break a later match, smart quotes that a finance system cannot parse, and combined codes like “Department — DEPT” that really want to be two columns. Power Query has a single-click button for all of these, and the whole chapter is about learning which button to reach for.

a. The practice file for this chapter

This chapter uses a single practice file:

5 - Shaping Data - Text Values.xlsx — a short list of conference delegates, exported from a system that was not especially careful with whitespace or case. About 40 rows and four columns: Full Name, Email, Department, Role.

The file is deliberately short so that the effect of every transformation is visible in the Data Preview without scrolling. Do not be misled by the size — the same transformations scale to thousands of rows.

b. Looking at the mess

Before pressing any buttons, open the file in Excel and look at the Delegates table. You will see most of these problems on the same sheet:

	A	B	C	D
1	Full Name	Email	Department	Role
2	ava BENNETT	Ava.BENNETT@Glasgow.AC.UK	Arts - AR	staff
3	liamO'Connor	liam.oconnor@student.gla.ac.uk	Social Sciences - SS	Student
4	MIA patel	Mia.Patel@Glasgow.ac.uk	Science & Engineering - SE	STUDENT
5	noah Campbell	NOAH.CAMPBELL@glasgow.ac.uk	Medical, Vet & Life Sci - MVLS	staff
6	“ISLA” ross	isla.ross@Glasgow.AC.UK	Arts - ar	Student
7	lucas MURRAY	lucas.MURRAY@glasgow.ac.uk	Science & Engineering - SE	Staff
8	Sofia Walker	sofia.walker@STUDENT.gla.ac.uk	Social Sciences - ss	student
9	oliver Graham	Oliver.Graham@glasgow.ac.uk	Arts - AR	Staff
10	EMILY Scott	emily.scott@glasgow.ac.uk	Medical, Vet & Life Sci - mvls	Student
11	jacob taylor	Jacob.Taylor@glasgow.ac.uk	Science & Engineering - SE	staff
12	Amelia REID	AMELIA.reid@Glasgow.ac.uk	Social Sciences - SS	student
13	henry– Young	henry.young@glasgow.ac.uk	Arts - AR	Staff
14	charlotte hughes	Charlotte.HUGHES@glasgow.ac.uk	Science & Engineering - SE	STUDENT

- Some names have leading or trailing spaces (“ava BENNETT”).
- Some names have inconsistent case — all lower, all upper, mixed.
- A few cells have invisible characters in them. Tabs and other control characters get pasted in from PDFs and Word documents and are impossible to see.
- Some names contain curly quotes (“...”) instead of plain quotes ("..."), and one contains a fancy dash (–) instead of a plain hyphen.
- The Department column is a combined value — full department name, a dash, and a short code — but we would like the full name and the code to live in separate columns.
- The Role column is inconsistent in case (“staff”, “Staff”, “STAFF”, “ Staff”).

Connect to this table in Power Query and the chapter will fix each of those problems in turn, one button at a time.

Task: Open the file and connect

- 1 Open **5 - Shaping Data - Text Values.xlsx** from your Practice Files folder.
- 2 Click anywhere inside the **Delegates** table.

- On the **Data** ribbon, click **From Table/Range**.
- The **Power Query Editor** opens. Leave the **Applied Steps** pane visible on the right — it will grow as you go.

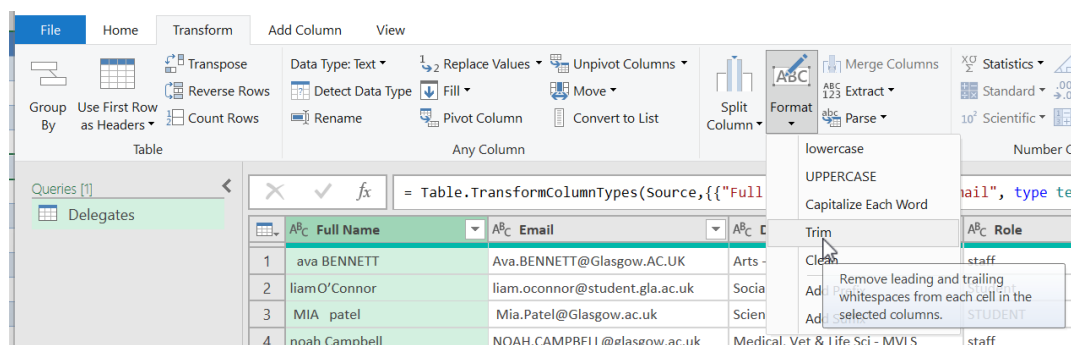
c. Trim — removing surrounding whitespace

Trim removes spaces from the start and end of every value in the selected column. Spaces in the middle of a value are left alone.

This is the single most common text transformation. Trailing spaces are almost invisible when you look at a cell, but they are the reason a VLOOKUP or a Merge fails even though the values “look” identical. Trim them as early as possible.

Task: Trim the Full Name column

- Click the **Full Name** column header to select the whole column.
- On the **Transform** ribbon, click **Format** → **Trim**.



- The leading and trailing spaces disappear from each value in the column.
- A new **Trimmed Text** step appears in **Applied Steps**.

Now do the same for **Email**, **Department**, and **Role**. You can trim columns one at a time or, if you want, select all four columns first (click the first header, then Ctrl+click the others) and trim them all in one click.

Task: Trim the remaining text columns

- Click the **Email** header, then Ctrl+click the **Department** header, then Ctrl+click the **Role** header.
- On the **Transform** ribbon, click **Format** → **Trim**.
- Leading and trailing spaces disappear from all three columns in one go. A single Trimmed Text1 step (or similar name) is added to Applied Steps.

Trim in Power Query is slightly stricter than Excel's TRIM function. Excel's TRIM also collapses internal double-spaces to single-spaces; Power Query's Trim does not. If internal double-spaces are an issue, use **Replace Values** (section f) to collapse them afterwards.

d. Clean — removing invisible characters

Clean strips out the invisible control characters that **Trim** does not touch. Tabs, line breaks, form feeds — the kind of debris that arrives with text pasted out of a PDF or an email. You cannot see these characters in the Data Preview, but they break comparisons and they break Split Column.

Clean is cheap and safe to run as a matter of habit on any text column that came in from an outside source.

Task: Clean all text columns

- 1 Select all four text columns (click the first, Ctrl+click the rest).
- 2 On the Transform ribbon, click **Format** → **Clean**.
- 3 The preview may look almost unchanged — that is expected, because the characters **Clean** removes are invisible in the first place.
- 4 A new **Cleaned Text** step appears in **Applied Steps**.

A rule of thumb: apply **Trim** and **Clean** together on text columns coming in from outside your organisation. They remove the two most common kinds of invisible text mess.

e. Change Case

The Format menu on the Transform ribbon offers three case transformations:

lowercase. Every letter becomes lower case. Useful for email addresses — email is not case-sensitive, so forcing lowercase makes duplicate-checking reliable.

UPPERCASE. Every letter becomes upper case. Useful for short codes where upper case is the convention (department codes, airport codes, room numbers).

Capitalize Each Word. The first letter of every whitespace-separated word becomes upper case; the rest become lower case. Useful for names, place names, and titles.

Task: Apply the right case to each column

- 1 Select the **Full Name** column. On the **Transform** ribbon, click **Format** → **Capitalize Each Word**.
- 2 Select the **Email** column. On the **Transform** ribbon, click **Format** → **lowercase**.
- 3 Select the **Role** column. On the **Transform** ribbon, click **Format** → **Capitalize Each Word**.
- 4 Leave the **Department** column alone for now — **Split Column** will do the job there.
- 5 Three new steps (Capitalized Each Word, Lowercased Text, Capitalized Each Word) appear in Applied Steps.

Capitalize Each Word is unaware of English naming conventions. It will turn “mcCallum” into “Mccallum” and “O’Connor” into “O’connor”. If you need MacCallum-style capitalisation there is no built-in fix — you would drop down to Replace Values (next section) for the handful of cases, or leave the column as text and let learners correct it in Excel after the load.

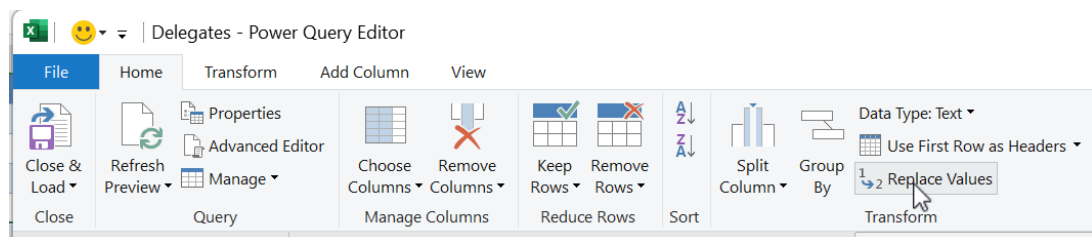
f. Replace Values

Replace Values does exactly what its name suggests: find every occurrence of one value in a column and replace it with another. It works on text, numbers, or dates. For text columns it is an exact-match replacement by default, but an Advanced option opens up “match entire cell contents” and “case-sensitive” toggles if you need them.

The classic use for Replace Values is fixing imported curly quotes, fancy dashes, or house-style replacements like changing “&” to “and” or “UofG” to “University of Glasgow”.

Task: Replace curly quotes with straight ones

- 1 Select the **Full Name** column.
- 2 On the **Home** ribbon, click **Replace Values**.



- 3 In **Value To Find**, paste or type the curly open-quote character: “
- 4 Leave **Replace With** empty.
- 5 Click **OK**. A new **Replaced Value** step is added to **Applied Steps**.
- 6 Repeat for a apostrophe (’), and an dash (-).

Each Replace Values button click adds its own step to the Applied Steps pane. If you prefer a tidier list, you can apply the replacements in a single shot using Replace Values on the Transform ribbon (which replaces within the current column selection) — but the separate steps are often easier to read when a query needs explaining later.

g. Split Column

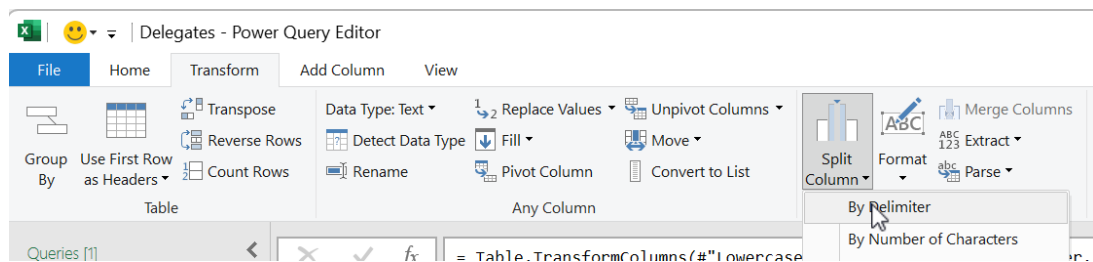
The Department column holds values that look like “Arts — AR” — a human-readable name, a delimiter, and a short code. This is one combined column in the source, but for analysis we usually want it as two columns.

Split Column on the Transform ribbon does this automatically. It asks for the delimiter to split on and, optionally, for how many times to split. It creates as many new columns as it needs.

Task: Split the Department column into name and code

- 1 Select the **Department** column.

- 2 On the **Transform** ribbon, click **Split Column** → **By Delimiter**.



- 3 Power Query inspects the column and offers a delimiter suggestion. Change the Select or enter delimiter dropdown to Custom.
- 4 Type a hyphen with a space either side: -
- 5 Under **Split at**, choose **Each occurrence of the delimiter**.
- 6 Click **OK**.
- 7 The column becomes two: **Department.1** (the name) and **Department.2** (the code).
- 8 A **Split Column by Delimiter** step and a **Changed Type1** step are added to **Applied Steps**.

Rename the two new columns so their meaning is clear. Double-click a column header and type the new name, then press Enter.

Task: Rename the split columns

- 1 Double-click the **Department.1** header. Type **Department** and press Enter.
- 2 Double-click the **Department.2** header. Type **Dept Code** and press Enter.
- 3 Two new **Renamed Columns** steps appear in **Applied Steps**.

Watch the Dept Code values — some of the originals had the code in lowercase (“Arts - ar”). Select Dept Code and apply Format → UPPERCASE to normalise them.

Task: Upper-case the Dept Code column

- 1 Click the **Dept Code** header.
- 2 On the **Transform** ribbon, click **Format** → **UPPERCASE**.
- 3 The codes all become consistent: AR, MVLS, SE, SS.

h. Loading the cleaned result

By this point every cell in the table is consistent. The column list is Full Name, Email, Department, Dept Code, Role — five tidy text columns. Load the result as a Table so you can see the effect of the work.

Task: Load the cleaned delegates table

- 1 On the **Home** ribbon, click the top half of **Close & Load** (not the arrow).
- 2 The Editor closes and Excel creates a new worksheet containing the cleaned delegates as a Table.
- 3 Compare the new sheet to the original **Delegates** sheet. Same data, but tidier, and the **Department** column is now two columns.

If you want to see a useful thing, go back to the original Delegates sheet and edit one of the values (for example, add four trailing spaces to a name). Then, on the Data ribbon, click Refresh All. The cleaned sheet will reflect your change — with the trailing spaces trimmed off — without you having to re-run any of the transformations.

Next: Chapter 6 moves from cleaning values to using them. Once a table is tidy, sorting and filtering become straightforward, and that is the subject of the next chapter.

6 Filtering and Sorting

Chapter 4 reshaped the structure of the data, and Chapter 5 cleaned up text values. Now that the data is tidy, we can start asking questions of it: show me the no-shows, sort by date, limit the result to a single college. Filtering and sorting are the most-used transformations in Power Query in day-to-day work, because they are the ones closest to the questions people ask.

Everything in this chapter happens from the header of a column. There is no separate ribbon for filter and sort; the buttons live beside each column, on the small drop-down arrow that appears when you click the header.

a. The practice file for this chapter

This chapter uses a single practice file:

6 - Filtering and Sorting.xlsx — the full ~5,000-row Conference Bookings dataset, in a Table called Bookings on a sheet called Bookings. Six columns: Booking ID, Student ID, Session Name, Session Date, Attendance, Fee.

Unlike Chapter 5, the file for this chapter is the real-sized dataset. Filtering and sorting are operations that matter most on volumes too big to do by hand — a 40-row file would make the tools look unnecessary.

Task: Open the file and connect

- 1 Open **6 - Filtering and Sorting.xlsx** from your Practice Files folder.
- 2 Click anywhere inside the **Bookings** table.
- 3 On the **Data** ribbon, click **From Table/Range**.
- 4 The Power Query Editor opens with the full dataset in the Data Preview.
- 5 A default Changed Type step appears in Applied Steps — this is normal. Leave it alone.

b. Filter and sort buttons on the column header

Each column header in the Data Preview has a small downward-pointing arrow on its right-hand edge. Click it and a dropdown menu appears. The top of that menu always offers Sort Ascending and Sort Descending. The middle offers a list of filter-by-condition options specific to the column's data type. The bottom offers a tick-list of the values actually present in the column, with a Search box and (Select All) at the top.

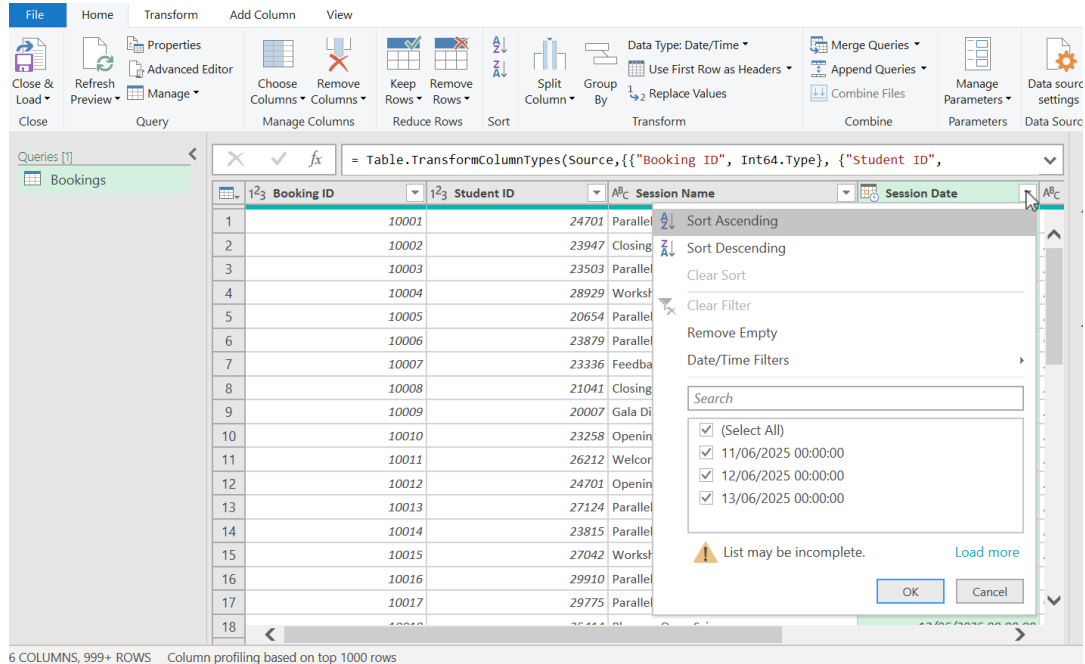
Every sort or filter applied from this dropdown becomes one step in the Applied Steps pane — just like every other transformation in the Editor.

c. Sort on a single column

The simplest case: sort ascending on Session Date, so that the earliest bookings come first.

Task: Sort by Session Date ascending

- 1 Click the small arrow on the right of the **Session Date** column header.



- 2 Choose **Sort Ascending**.
- 3 The **Session Date** column acquires a small 1↑ indicator on its header, showing that it is the first (and only) sort column.
- 4 A new **Sorted Rows** step appears in **Applied Steps**.

Sort Descending works the same way and, as expected, does the opposite. You can only have one kind of sort active on a column at a time — clicking Sort Ascending and then Sort Descending replaces the first sort rather than adding a second one.

d. Sort on multiple columns

Real questions often ask for more than one sort column. “Sort by Session Date, then within each date by Session Name” — this is a two-level sort. Power Query handles this in a natural way: after sorting on one column, sort on a second column, and Power Query shows both sort orders with numbered arrows on the column headers.

Task: Add Session Name as a secondary sort

- 1 The preview is already sorted by **Session Date** ascending from the previous task.
- 2 Click the arrow on the **Session Name** column header.
- 3 Choose **Sort Ascending**.
- 4 The Session Date indicator changes to 1↑ and Session Name acquires 2↑. The preview is now sorted first by date, then by session name within each date.

- 5 Notice the Applied Steps pane: a second Sorted Rows step has not been added — Power Query has edited the existing Sorted Rows step to include the second sort key.

Power Query's sort is stable. Ties in the primary sort key are broken by the secondary sort, ties in the secondary are broken by the tertiary, and so on. This is the same behaviour as Excel's Sort dialog with multiple levels.

To remove the secondary sort, click the Session Name column arrow and choose Clear Sort.

e. Filter by value (tick-list filter)

The tick-list filter is the bottom of the column dropdown. Power Query lists every distinct value in the column, and you tick the ones you want to keep. This is the same interface Excel uses for AutoFilter, and most learners will have used it before.

One caution specific to Power Query: the tick-list only shows the values that appear in the first 1,000 rows of the preview. If you expect a value to be in the list and cannot find it, type it into the Search box at the top of the dropdown rather than scrolling. The Search box looks at the full data, not just the preview.

Task: Filter to show only no-show bookings

- 1 Click the arrow on the **Attendance** column header.
- 2 In the tick-list at the bottom, click (**Select All**) once to untick all values.
- 3 Tick only **No-show**, then click **OK**.
- 4 The preview is reduced to only the bookings where the delegate did not turn up.
- 5 A new **Filtered Rows** step appears in **Applied Steps**. The **Attendance** column header now shows a small funnel icon, indicating an active filter.

To remove the filter, click the funnel icon on the **Attendance** header and choose Clear Filter. The Filtered Rows step will be removed from Applied Steps automatically.

f. Filter by condition

Filtering by condition is more flexible than the tick-list. Instead of ticking a list of values, you describe what you want — “before this date”, “greater than this amount”, “contains this text” — and Power Query keeps the matching rows.

The middle of the column dropdown changes name based on the column type: Text Filters, Number Filters, or Date Filters. Each contains a sub-menu of the conditions you would expect.

Task: Show only bookings with a fee of £10 or more

- 1 Click the arrow on the **Fee** column header.
- 2 Choose **Number Filters** → **Greater Than Or Equal To...**
- 3 In the dialog, type **10** in the **Value** field and click **OK**.

- 4 The preview now contains only bookings with a fee of £10 or higher. A second **Filtered Rows** step is added to **Applied Steps** (the no-show filter from the previous task is still in place).
- 5 The **Fee** column header now also has a funnel icon.

Task: Show only Workshop sessions

- 1 Click the arrow on the **Session Name** column header.
- 2 Choose **Text Filters** → **Begins With...**
- 3 Type **Workshop** in the field and click OK.
- 4 The preview now contains only bookings whose session name starts with “Workshop”.
- 5 A third **Filtered Rows** step appears in **Applied Steps**.

Text filters in Power Query are case-insensitive by default. If case matters (e.g. distinguishing “Ph.D” from “PhD”), you would need to drop down to an Advanced filter. That is beyond the scope of this course.

Number Filters, Text Filters, and Date Filters each have a short list of the most common conditions as direct menu items, plus a Custom Filter... option that opens a two-condition dialog (“equals X and is greater than Y”, and so on).

g. Removing and editing filter/sort steps

Every filter and every sort in Power Query is a step — exactly like every other transformation. To undo one, select the step in the Applied Steps pane and press Delete, or hover over the step and click the small X that appears. The preview will re-run the remaining steps and update.

To edit a filter condition after the fact — say, to change the fee threshold from 10 to 25 — click the small gear icon beside the step in the Applied Steps pane. The original dialog reopens so you can change the value.

Task: Edit the fee threshold step

- 1 In the **Applied Steps** pane, find the **Filtered Rows** step that represents the Fee ≥ 10 filter (hover over each step to see its definition as a tooltip).
- 2 Click the small gear icon to the right of the step name.
- 3 The **Greater Than Or Equal To** dialog opens with 10 already filled in. Change it to 25 and click **OK**.
- 4 The preview re-runs and now shows only bookings with a fee of £25 or more.
- 5 Chapter 10 will cover the Applied Steps pane in more detail — renaming, reordering, and deleting steps. This chapter just introduces the idea that the steps are editable.

h. Loading the filtered, sorted result

At this point the query contains three filters and one two-level sort, expressed as a small number of steps in the Applied Steps pane. Load the result as a Table into Excel so that you have a tidy, filtered view of the bookings to work with.

Task: Load the filtered result

- 1 On the **Home** ribbon, click the top half of **Close & Load**.
- 2 The Editor closes and Excel creates a new worksheet containing the filtered, sorted rows as a Table.
- 3 The **Queries & Connections** pane on the right shows a query named **Bookings** with a row count — this is the number of rows that survived the filters.

If the source workbook later gets more rows added to the Bookings table (for example, new registrations), clicking Refresh All on the Data ribbon will re-run the query — every filter and sort will be re-applied, and the output table will update to match.

Next: Chapter 7 adds new columns to the table — columns that are calculated from the existing ones, using three different techniques.

7 Adding a Calculated Column

Filtering and sorting in Chapter 6 reshaped the rows of the table. This chapter goes the other way: it keeps the rows exactly as they are and adds new columns, each derived from the columns that already exist. If you have written formulas in Excel — SUM, IF, CONCAT — calculated columns will feel familiar. Power Query provides three separate tools for building them, each aimed at a different kind of job.

All three tools live on the Add Column ribbon in the Power Query Editor.

a. The practice file for this chapter

This chapter uses a single practice file:

7 - Adding a Calculated Column.xlsx — a clean copy of the full ~5,000-row Conference Bookings dataset. Table is called Bookings, on a sheet called Bookings. Six columns: Booking ID, Student ID, Session Name, Session Date, Attendance, Fee.

Task: Open the file and connect

- 1 Open **7 - Adding a Calculated Column.xlsx** from your Practice Files folder.
- 2 Click anywhere inside the **Bookings** table.
- 3 On the **Data** ribbon, click **From Table/Range**.
- 4 The Power Query Editor opens. An automatic Changed Type step appears in Applied Steps.

b. Three ways to add a column

Power Query gives you three tools for adding a calculated column. Each of them produces the same kind of result — a new column whose values are derived from the existing rows — but they take very different inputs.

Column from Examples. You type in what you want the value in the new column to be for two or three example rows. Power Query inspects your examples, guesses the rule, and fills the rest of the column automatically. Best for text transformations that are easy to show and hard to describe.

Custom Column. You write a small formula using M, the language behind Power Query. Best for numeric calculations and anything that combines values from multiple columns.

Conditional Column. You build a table of if-then rules through a dialog. Best for categorising rows (“if Attendance is Attended then Present, else ...”).

All three add a step to the **Applied Steps** pane named Added Custom, Inserted Text, or something similar — just like every other transformation. And like every other step, they can be edited, deleted, or reordered later.

c. Column from Examples

Column from Examples is the easiest of the three to use, and the most surprising the first time you see it work. It asks you to type what you want — not how to calculate it — and it infers the transformation from the examples you give.

Good candidate columns for this tool: extracting a first name or last name from a full name, pulling the year out of a date, shortening a session title. Anything where, if you were describing it in words, you would say “the part before the colon” or “just the month name”.

The tool works best when you give it two or three examples, picked from rows that look different from each other. One example is often enough — but ambiguous cases may need more.

Task: Use Column from Examples to get a short session name

- 1 On the **Add Column** ribbon, click **Column from Examples** → **From Selection**.
- 2 Click the **Session Name** column header tick box so that Power Query knows you are working from that column.
- 3 A new blank column appears on the right, with a heading cell and an empty column beneath it.
- 4 In the first cell of the new column, type Digital Humanities and press Enter. (This is the part after the colon in “Parallel B: Digital Humanities”).
- 5 Power Query may immediately fill down a suggestion for every other row. If it gets a row wrong, double-click that cell and type the right answer, then press **Enter**.
- 6 Once every visible row looks correct, click OK.
- 7 A new column appears with a name like “**Text After Delimiter**” or similar, and an Inserted Text step is added to **Applied Steps**.

- 8 Rename the new column: double-click the header and type Short Session, then press Enter.

If Power Query cannot infer a rule, the new column stays empty after the first example. Try a different example — one where the rule is clearer — or add a second and third example until the rest of the column fills in. Two or three examples is enough for any transformation this tool can handle.

d. Custom Column

Custom Column opens a small dialog where you write an M formula. M is the language Power Query runs under the hood; every button you click in the Editor is quietly generating M in the background. For this chapter, you will write only the simplest kind of M formula: an arithmetic expression that combines existing column values.

The syntax you need for now: refer to a column by wrapping its name in square brackets, e.g. [Fee]. Use + – * / for arithmetic. Use & to concatenate text. Number and text literals follow the same rules as Excel — numbers plain, text in double quotes.

Task: Add a Fee Including VAT column

- 1 On the **Add Column** ribbon, click **Custom Column**.
- 2 In the New column name field, type Fee inc VAT.
- 3 In the Custom column formula field, type: [Fee] * 1.2
- 4 (Notice that the field on the right lists every available column — you can double-click a column name to insert it.)
- 5 A message at the bottom of the dialog reads “No syntax errors have been detected.” If it says anything else, check your brackets.
- 6 Click OK.
- 7 A new Fee inc VAT column appears with values that are each the original Fee multiplied by 1.2. An Added Custom step is added to Applied Steps.
- 8 Check the data type icon on the new column header. Power Query will often set it to ABC123 (Any). Click the icon and change it to Decimal Number.

That was a pure numeric formula. Custom Column is equally happy combining text. As a second demo, build a label that includes both the session name and the date in one column.

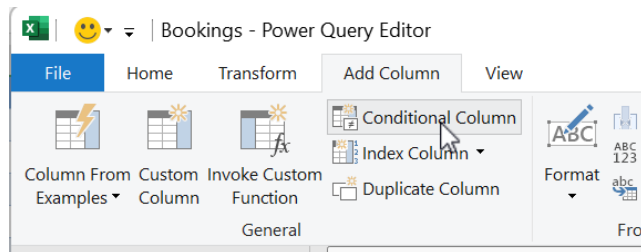
e. Conditional Column

Conditional Column is the tool to reach for when you want to categorise rows. It lets you build a table of if-then rules through a dialog, without writing any M at all. The rules are evaluated top-to-bottom; the first one that matches wins. The last row is always an else clause that catches anything the earlier rules did not.

The classic use is turning a raw column of values into a tidy category column. The Attendance column in the bookings data has three distinct values — Attended, No-show, and Cancelled — and we will use Conditional Column to produce a simpler two-way split: Present or Absent.

Task: Add a Present/Absent category column

- 1 On the **Add Column** ribbon, click **Conditional Column**.



- 2 In **New column name**, type **Present**.
- 3 In the first rule row: set Column Name to **Attendance**, Operator to **equals**, Value to **Attended**. Set the Output to **Present**.
- 4 Click **Add Clause** to add a second rule row: Column Name **Attendance**, Operator **equals**, Value **No-show**, Output **Absent**.
- 5 Click **Add Clause** a third time: **Attendance** equals **Cancelled**, Output **Absent**.
- 6 In the **Else** field at the bottom, type **Unknown**. (This catches anything unexpected that arrives in the column in future.)

A screenshot of the 'Add Conditional Column' dialog box in Power Query. The 'New column name' is 'Present'. There are three rule rows defined:

- Row 1: If 'Attendance' equals 'Attended', Then 'Present'.
- Row 2: Else If 'Attendance' equals 'No-show', Then 'Absent'.
- Row 3: Else If 'Attendance' equals 'Cancelled', Then 'Absent'.

An 'Add Clause' button is below the rows. At the bottom, the 'Else' field is set to 'unknown'. 'OK' and 'Cancel' buttons are at the bottom right.

- 7 Click **OK**.
- 8 A new **Present** column appears showing **Present**, **Absent**, or **Unknown** for each row. An **Added Conditional Column** step is added to **Applied Steps**.

The Conditional Column dialog also supports rules that compare a column's value to another column, not just to a literal. Click the small column-icon button beside the Value field and the dialog lets you choose a column instead.

When you need logic more complex than Conditional Column can express — for example, “if Fee is between 10 and 25 AND Attendance is Attended” — drop back to Custom Column and use an if ... then ... else expression in M. Chapter 11 shows a small example.

f. Which tool to reach for

A rule of thumb for the three tools:

Text? Try Column from Examples first. It is astonishingly good at inferring text transformations, and getting the same result from M can be fiddly.

Number? Use Custom Column. The formula is usually one line of arithmetic.

Category? Use Conditional Column.

There is no wrong answer — any of the tools can do any of the jobs, in principle. But starting with the right one will save you time.

g. Loading the result

The query now contains three new columns on top of the original six. Load the result as a Table.

Task: Load the augmented table

- 1 On the **Home** ribbon, click the top half of **Close & Load**.
- 2 Excel creates a new worksheet with the full bookings table plus the three calculated columns: Short Session, Fee inc VAT, Session Label, and Present.
- 3 Scroll across to confirm all calculated columns are present and populated.

Next: Chapter 8 moves from modifying a single query to combining two queries. Append will be introduced as the way to stack two tables of the same shape on top of each other.

8 Appending Tables

Up to this chapter every task has involved a single query working from a single source. The next two chapters change that. Chapter 8 shows how to combine two tables of the same shape by stacking one on top of the other — that is Append. Chapter 9 shows how to combine two tables of different shapes by matching rows through a common key — that is Merge.

Appending is the simpler of the two, and the one you reach for most often. A month-by-month sales file, a set of quarterly returns, two quarters of bookings — any situation where the same kind of rows are spread across separate files — is an Append job.

a. The practice files for this chapter

This chapter uses two practice files, one for each half of the bookings:

8a - Appending Tables - Early Bookings.xlsx — bookings from 11 and 12 June 2025, inside an Excel Table called Early.

8b - Appending Tables - Late Bookings.xlsx — bookings from 13 June 2025, inside an Excel Table called Late.

The two files contain the same kind of data (bookings for the conference), but they were exported at different times and have a small problem we will need to notice and fix. The 8b file has its columns in a different order, and one header is spelled differently: Fee (GBP) instead of Fee. This is a deliberately realistic detail — two different admin systems exported the same idea with different column shapes.

Close any earlier Power Query Editor windows before you start. This chapter will create multiple queries in a single workbook, and it is easier to follow the Queries pane when you start with it empty.

b. What Append means

Append takes two or more queries that contain rows of the same kind and produces a new query whose rows are the rows of the first, followed by the rows of the second (followed by the third, and so on). If both source queries have a column called Booking ID, the appended result has a single Booking ID column. If one source has a column the other does not, that column still appears in the result — rows from the table that did not have it simply hold the value null.

This is purely a row-stacking operation. Nothing is matched, compared, or related; the two queries do not have to share any common value, and in fact the order of the rows in the appended result is simply the order of the first source followed by the order of the second source.

c. Load both source tables as Connection Only

To Append two queries you first need to have two queries. The workflow is always the same: connect to each source and load it as a Connection Only, then do the Append using the two Connection-Only queries as inputs.

You have already seen Connection-Only loading in Chapter 3. Here you will do it twice in the same workbook so that both sources are available to Append.

Task: Create a connection to the Early bookings

- 1 Open an empty new Excel workbook. This is the workbook that will hold the combined result.
- 2 On the **Data** ribbon, click **Get Data** → **From File** → **From Workbook**.
- 3 Browse to **8a - Appending Tables - Early Bookings.xlsx** and click **Import**.
- 4 In the **Navigator** dialog, click the **Early1** table (not the Early sheet) so the preview shows the data.
- 5 Click **Transform Data**. (Do not click Load — the goal is to create a Connection.)
- 6 The Editor opens. Nothing needs changing. On the Home ribbon, click the arrow below Close & Load and choose **Close & Load To...**
- 7 In the **Import Data** dialog, choose **Only Create Connection** and click **OK**.
- 8 A query called Early1 now exists in the Queries & Connections pane on the right, marked Connection Only.

Task: Create a connection to the Late bookings

- 1 Repeat the steps above for the second file: **Get Data** → **From File** → **From Workbook**, browse to **8b - Appending Tables - Late Bookings.xlsx**, select the **Late1** table in the **Navigator**, and click **Transform Data**.
- 2 Inside the Editor, have a quick look at the **Late** table. Note that the column order is different from the Early table (Session Date is in the second position) and that the Fee column is named Fee (GBP).
- 3 Without making any other changes, close the Editor via Close & Load To... → Only Create Connection.
- 4 A second query called Late now appears in Queries & Connections, also marked Connection Only.

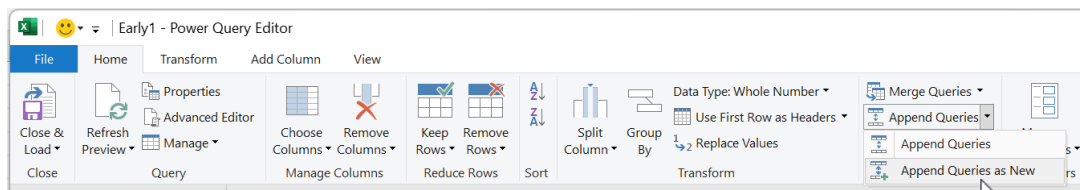
At this point the Excel workbook still looks empty — no sheet has data on it. That is correct. Both queries exist only as connections; their data lives in the Power Query engine, not in the worksheet.

d. Append the two tables

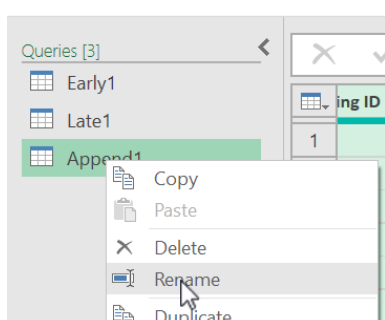
Now that both source tables are available as queries, you can build a third query that appends them. Append lives on the Home ribbon of the Editor.

Task: Append Early and Late into a new query

- 1 In **Queries & Connections**, double-click **Early** to open it in the Editor.
- 2 On the **Home** ribbon, click the bottom half of **Append Queries** → **Append Queries as New**. (The two options do slightly different things — see the note below.)



- 3 The **Append** dialog opens. Leave Two Tables selected. For the Primary Table, confirm it is Early. For the Table to Append, choose Late from the dropdown.
- 4 Click **OK**.
- 5 A new query appears in the Queries pane, called **Append1** or similar. The preview contains the rows from Early at the top and the rows from Late below.



- 6 Rename the new query: right-click Append1 in the Queries pane and choose Rename. Type **All Bookings** and press **Enter**.

Append Queries. Modifies the current query in place — it adds an Appended Query step to the Applied Steps pane of whichever query you had open. Use this when the appended result should replace the primary query.

Append Queries as New. Creates a brand-new query whose only step is the Append. Use this (as we did above) when you want both of the original Connection-Only queries to remain as-is, and the append to live in a new, separate query. This is almost always what you want.

e. Headers that do not match

Look at the appended result carefully. Scroll across to the right — there should be seven columns, not six. Two of them are suspiciously similar:

Fee — populated for rows that came from Early, null for rows that came from Late.

Fee (GBP) — null for rows that came from Early, populated for rows that came from Late.

= Table.Combine({Early1, Late1})						
	Session Name	Session Date	Attendance	Fee	Fee (GBP)	
1	24701 Parallel B: Digital Humanities	45819	No-show		0	
2	23503 Parallel B: Digital Humanities	45819	Attended		0	
3	22870 Parallel B: Culture & Identity	45820	Cancelled		0	

This is Append working exactly as documented. It matches columns by header text — not by position. Fee and Fee (GBP) are, as far as the tool can tell, unrelated columns. The tool

kept them separate and filled in null where a source did not have the corresponding header.

This is the most important lesson in the chapter. When you Append, always look for mismatched columns in the result. If two columns look like they should have been one, the header text of one of the sources needs changing.

f. Fixing the mismatch

The fix is to rename the Fee (GBP) column in the Late query to just Fee, so that the Append matches them on the way through. You do this by editing the Late query — not by editing the appended result.

Task: Rename the column in the Late query

- 1 In the **Queries** pane, double-click **Late** to open it in the Editor.
- 2 Double-click the **Fee (GBP)** column header. Type **Fee** and press **Enter**.
- 3 A Renamed Columns step appears in the Applied Steps pane of the Late query.
- 4 Now double-click **All Bookings** to reopen it. The preview should automatically re-run; there is now only one **Fee** column and it is populated for every row.
- 5 Scroll across and confirm: six columns, every row populated, earlier dates at the top, later dates at the bottom.

This is the thing that makes Power Query so useful. The fix lives in the Late query. The Append query did not need to be touched. A month later, when a new batch of data arrives and you re-run the pipeline, the rename is still in place.

If you ever see a “mystery null column” in an appended result, the cause is almost always a header typo or a renamed-away column in one of the sources. Walk backwards: open the source query and look for the offending header.

g. Loading the appended result

Now that the appended result is a clean single table, load it.

Task: Load All Bookings

- 1 Ensure **All Bookings** is the query open in the Editor.
- 2 On the **Home** ribbon, click the top half of **Close & Load**.
- 3 Excel creates a new worksheet containing the combined bookings as a Table.
- 4 The **Queries & Connections** pane shows three queries: **Early** (Connection Only), **Late** (Connection Only), and **All Bookings** (loaded). The total row count on **All Bookings** should equal the sum of the Early and Late row counts.

If you ever need to add a third month, create another Connection-Only query for it and edit the Append step in All Bookings to include it. Power Query Append supports any number of input queries, not just two.

Next: Chapter 9 introduces Merge, which is Append's more powerful cousin. Where Append stacks rows, Merge matches rows between two differently-shaped tables using a common key.

9 Merging Tables

Chapter 8 stacked two tables of bookings on top of each other to give a single, longer table. Append is the right tool when both inputs contain the same kind of row. This chapter covers the other case: two tables of different kinds, where each booking row needs to be enriched with information about the student who made it. The booking table has a Student ID; a separate Student List table has the student's First Name and College beside their ID. Bringing those two together is what Merge does.

If you have ever written a VLOOKUP, INDEX/MATCH, or XLOOKUP in Excel, Merge is doing the same job. The difference is that Merge does it as a recorded step in the query rather than as a formula stretched down a column. Set it up once; refresh runs it again.

a. The practice files for this chapter

This chapter uses two practice files:

- 9a - Merging Tables - Bookings.xlsx — the full ~5,000-row Conference Bookings dataset.
- 9b - Merging Tables - Student List.xlsx — the Student List, ~500 rows. Three columns: Student ID, Student First Name, College.

As with Chapter 8, start with a fresh new empty workbook. Both source files will be loaded into it as Connection-Only queries, then merged.

b. What Merge means (and how it differs from Append)

Append. “Take all rows from table A and stick all rows from table B underneath them. Match by header.”

Merge. “For each row in table A, find the matching row in table B (by some shared column), and bring some of B's columns alongside.”

Merge does not change the number of rows in the primary table (in the most common case). It changes the number of columns: the primary table comes out with one or more new columns drawn from the secondary table.

The shared column — the one whose values are used to find the match — is called the join key. Both tables must contain a column that means the same thing. In our case the join key is Student ID, and it is named Student ID in both files. (If the names had been different, Merge would still work — but you would have to choose the matching columns by hand in the dialog.)

c. Load both source tables as Connection Only

Same workflow as Chapter 8. Two queries, each marked Connection Only.

Task: Connect to the Bookings file

- 1 Open a new empty Excel workbook.
- 2 On the **Data** ribbon, click **Get Data** → **From File** → **From Workbook**.
- 3 Browse to **9a - Merging Tables - Bookings.xlsx** and click **Import**.
- 4 In the **Navigator**, click the **Bookings** table and click **Transform Data**.

- 5 The Editor opens. No transformations are needed. On the **Home** ribbon, click the arrow below **Close & Load** and choose **Close & Load To... → Only Create Connection**.
- 6 A query called **Bookings** now exists in **Queries & Connections**, marked **Connection Only**.

Task: Connect to the Student List file

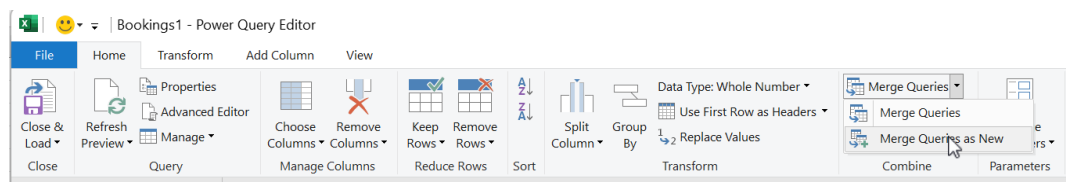
- 1 Repeat for the second file: **Get Data → From File → From Workbook → 9b - Merging Tables - Student List.xlsx**.
- 2 In the Navigator, choose the **Student List** table and click **Transform Data**.
- 3 Inside the Editor, glance at the columns: Student ID, Student First Name, College.
- 4 Close the Editor with **Close & Load To... → Only Create Connection**.
- 5 A second query, called **Student List**, appears in **Queries & Connections**.

d. Merge Bookings with Student List

Now you have the two queries you need. Merge lives on the Home ribbon.

Task: Run the Merge

- 1 In the **Queries** pane, double-click **Bookings** to open it in the Editor.
- 2 On the **Home** ribbon, click the bottom half of **Merge Queries → Merge Queries as New**. (As with Append, the as New variant is the safer default — it leaves the original Bookings query untouched and creates a third query for the merged result.)



- 3 The Merge dialog opens. The top half is the primary table; the bottom half is the secondary table.

×

Merge

Select tables and matching columns to create a merged table.

Bookings1

Booking ID	Student ID	Session Name	Session Date	Attendance	Fee
10001	24701	Parallel B: Digital Humanities	45819	No-show	0
10002	23947	Closing Keynote	45821	Attended	0
10003	23503	Parallel B: Digital Humanities	45819	Attended	0
10004	28929	Workshop: Publishing Your First Paper	45821	Attended	15
10005	20654	Parallel B: Engineering & Society	45821	Attended	0

Student List

Student ID	Student First Name	College
20007	Mateo	Science & Engineering
20010	Niamh	Science & Engineering
20028	Xin	Medical, Veterinary & Life Sciences
20036	Ivy	Science & Engineering
20054	Jane	Social Sciences

Join Kind

Left Outer (all from first, matching from second)

☐ Use fuzzy matching to perform the merge

▸ Fuzzy matching options

OK

Cancel

- 4 In the top half, the table is already shown as **Bookings**. Click the **Student ID** column header in the top preview to mark it as the join key on the primary side.
- 5 In the bottom half, choose **Student List** from the dropdown. The preview appears.
- 6 Click the **Student ID** column header in the bottom preview to mark it as the join key on the secondary side.
- 7 At the bottom of the dialog, look at the **Join Kind** dropdown. Leave it on **Left Outer (all from first, matching from second)**. The note below the dropdown reads something like “5,065 rows match in 5,065 of 500 ...” — confirming that every booking has a corresponding student.
- 8 Click **OK**.
- 9 A new query (Merge1 or similar) appears in the Queries pane.
- 10 Rename it: right-click Merge1, choose Rename, type **Bookings with Students**, and press **Enter**.

In the Editor preview, you will see the original six Bookings columns plus a new column on the right called Student List. Each cell in that column shows the word Table in green text — it is a placeholder for the row in Student List that matched this booking.

e. Choosing the join kind

The Join Kind dropdown has six options. Each one decides what to do when a row on one side has no match on the other side. Most everyday tasks use one of the top two:

Left Outer (all from first, matching from second). Every row from the primary (left) table appears in the result, with values from the secondary (right) table where there is a match and null where there is not. This is the same behaviour as VLOOKUP. The default and the most-used option.

Inner (only matching rows). Only rows where the primary table has a match in the secondary table appear. Rows that did not match are dropped. Useful for cleaning unmatched rows out of the data.

The other four (Right Outer, Full Outer, Left Anti, Right Anti) cover specialised cases. They are named for the same join kinds you see in SQL. For this course, Left Outer is the right answer for any “lookup” job and Inner is the right answer for any “keep matches only” job.

Pay attention to the small status line in the Merge dialog (“The selection matches X of Y rows”). It tells you how many rows in the primary table found a match in the secondary table — a quick sanity check before you click OK. A surprise low number usually means the join key was set wrongly or the data types of the two key columns do not match.

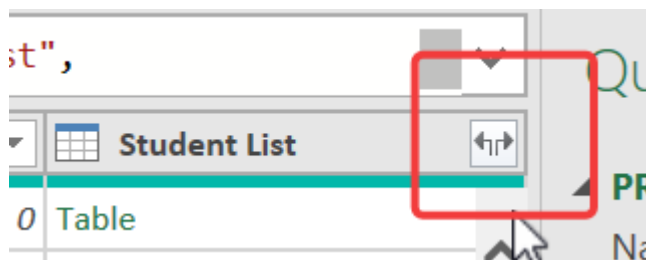
f. Expanding the merged column

After the Merge, the secondary table appears as a single column on the right of the preview, with the word Table in every cell. To use the columns inside (First Name, College), you need to expand the column.

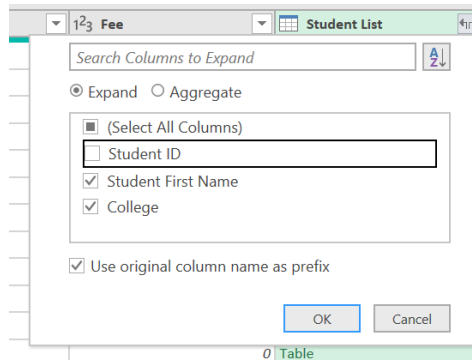
In the column header is a small icon showing two arrows pointing apart. Click it. A small panel drops down listing every column in the secondary table, with a tick-box beside each one.

Task: Expand the Student List column

- 1 Click the expand icon (two arrows) on the right edge of the Student List column header.



- 2 A panel appears listing **Student ID**, **Student First Name**, **College**.
- 3 Untick Student ID — you already have it from the Bookings side; pulling it in twice is redundant.



- 4 Leave **Student First Name** and **College** ticked.
- 5 At the bottom, untick the **Use original column name as prefix** box. (If left ticked, the new columns would be called Student List.Student First Name and Student List.College, which is unwieldy. Untick it for cleaner names.)
- 6 Click **OK**.
- 7 The Student List column is replaced with two new columns: **Student First Name** and **College**. Every row is populated.
- 8 An **Expanded Student List** step is added to **Applied Steps**.

If you ever need to add another column from the secondary table later, edit the Expanded step (gear icon) and tick the additional column. You do not need to redo the Merge.

g. Loading the merged result

Task: Load the merged table

- 1 On the **Home** ribbon, click the top half of **Close & Load**.
- 2 Excel creates a new worksheet containing the bookings with two extra columns (Student First Name, College) populated from the **Student List**.
- 3 The Queries & Connections pane now lists three queries: **Bookings** (Connection Only), **Student List** (Connection Only), **Bookings with Students** (loaded).

This is the same pattern as Append from Chapter 8: two source connections feeding into a third query that does the combining. As before, Refresh All on the Data ribbon will re-run the whole pipeline if either of the source files is updated.

Next: Chapter 10 takes a step back from new transformations and looks more carefully at the Applied Steps pane itself — renaming, reordering, deleting, and editing the steps that have been quietly accumulating throughout the course.

Useful Shortcut keys

Using keyboard shortcuts can help you become more efficient when creating documents in Microsoft applications. Most keyboard shortcuts require you to use two or more keys at the same time. To use a keyboard shortcut first press and hold down the modifier key or keys (i.e. SHIFT, CTRL, ALT) and then press the corresponding standard key on your keyboard.

Function	Shortcut
Go to "Tell me what you want to do"	ALT+Q
Open	CTRL+O
Save	CTRL+S
Close	CTRL+W
Cut	CTRL+X
Copy	CTRL+C
Paste	CTRL+V
Select all	CTRL+A
Bold	CTRL+B
Italic	CTRL+I
Underline	CTRL+U
Cancel	Esc
Undo	CTRL+Z
Re-do	CTRL+Y