EXCEL AND POWER BI REFERENCE

Power Query Cheat Sheet

A practical guide to getting data, shaping it, writing useful M, avoiding refresh pain, and building queries that survive real business files.

Excel: Get & Transform | **Power BI Desktop** | **Power Query Online** | **Updated: 2026-04-28**

Use this when you need to...

- Clean CSV, Excel, database, folder, web, SharePoint, API, and dataflow sources.
- Combine files, reshape columns, split messy text, unpivot reports, and standardize dates.
- Create repeatable refresh logic instead of manual copy/paste cleanup.
- Prepare tables before Excel PivotTables, Power Pivot, Power BI models, or DAX measures.

The mental model

Power Query records every transformation as an applied step. The UI writes M code for you. Refresh reruns the recipe from source to output.

Best habit: keep raw extraction, reusable staging, business logic, and final load tables separated.

What Power Query is good at

- ETL / ELT prep, row and column shaping, joins, appends, data type fixes.
- Reusable file ingestion and repeatable refresh.
- Reducing model size by removing unused rows and columns before load.

What it is not for

- Interactive calculations that must respond to report filters. Use DAX or Excel formulas.
- Row-by-row procedural scripting when a table operation exists.
- Heavy transformations that belong in SQL, a lakehouse, warehouse, or data pipeline.

Generated as a printable reference. Sources and further reading are listed at the end.

1. Fast Orientation

2. Where to Start

Excel

- **Open editor:** Data tab > Get Data, From Table/Range, or Queries & Connections > Edit.
- **Load options:** Close & Load To > Table, PivotTable Report, PivotChart, Only Create Connection, or Add this data to the Data Model.
- **Best for:** reusable spreadsheet cleanup, audit-friendly steps, Power Pivot inputs, small to medium extracts, and office workflows.
- **Watch:** workbook file size, local credentials, refresh settings, and accidental worksheet loads of staging queries.

Power BI Desktop

- **Open editor:** Home > Transform data, or Get data > Transform Data.
- **Apply:** Home > Close & Apply sends enabled-load queries into the semantic model.
- **Best for:** model-ready tables, star schemas, query folding, incremental refresh design, shared datasets, and scheduled refresh.
- **Watch:** load-disabled staging queries, DirectQuery limitations, privacy levels, gateway credentials, and model relationships.

Concept	Plain meaning	Practical advice
Query	A repeatable set of steps that returns a value, usually a table.	Name it like an output: `FactSales`, `DimCustomer`, `stg_SalesCsv`.
Applied step	A named M expression in the query pipeline.	Rename important steps; delete auto steps that do not help.
Preview	Sample data shown in the editor, not always the entire dataset.	Do not trust preview row counts for full refresh behavior.
Load	The destination after Power Query finishes.	Disable load for helper/staging queries in Power BI; use connection-only in Excel when appropriate.
M	The functional, case-sensitive language behind all steps.	Use Advanced Editor for exact control, parameters, functions, and reusable patterns.

> **Repeatable rule:** Power Query should create clean tables. DAX or Excel formulas should calculate analytics after the clean tables load.

3. The Golden Workflow

1. Connect
Source and navigate
Choose connector, object, file, folder, API, or query.

2. Reduce
Filter and select
Remove unused rows and columns as early as safely possible.

3. Clean
Standardize values
Trim, clean, replace, split, parse, and fix headers.

4. Shape
Model-ready table
Merge, append, group, pivot/unpivot, create keys.

5. Load
Typed final output
Set final data types, rename, reorder, and load only needed tables.

Recommended query layers

- **Raw:** source connection and navigation only. Rarely edited after creation.
- **Staging:** reusable cleanup, data type harmonization, and source-specific fixes. Disable load.
- **Transform:** joins, appends, business rules, surrogate keys, and filtering.
- **Final:** model-facing tables: facts, dimensions, report tables, or worksheet outputs.

Order of operations

1. Filter rows and select columns before expensive operations.
2. Fix column names before referencing them in later steps.
3. Set data types before merges and calculations that depend on type.
4. Put final type conversion near the end when files are messy or schema changes often.
5. Keep sorting for final presentation only; sort is often expensive and may not matter after load.

Naming

- `src_` source-only.
- `stg_` cleaned reusable base.
- `fn_` function.
- `p_` parameter.
- `Dim` , `Fact` model tables.

Applied step hygiene

- Delete redundant `Changed Type` steps.
- Rename unclear steps after important transformations.
- Keep `Source` and `Navigation` simple.
- Use comments in M for non-obvious business rules.

Load hygiene

- Disable load for staging/helper queries.
- Load only columns used by reports or downstream models.
- Prefer narrow fact tables and clean dimension tables.
- Validate row counts before publishing.

4. Interface Shortcuts and High-Value Buttons

Need	Where	Tip
View generated M	Home > Advanced Editor	Use it to fix column references, add variables, create functions, or remove bloated UI code.
Inspect one step	Applied Steps pane	Select a step and check whether the preview first becomes wrong there.
Formula bar	View > Formula Bar	Keep it on. It is the fastest way to learn M safely.
Column profile	View > Column quality / distribution / profile	Switch from top 1,000 rows to entire dataset when validating small or medium files.
Dependencies	View > Query Dependencies	Find staging queries, circular references, and accidental worksheet outputs.
Native query	Right-click applied step > View Native Query	If available, folding is still happening up to that step.
Diagnostics	Tools > Start Diagnostics	Use for slow refreshes; compare before and after reducing columns or moving filters earlier.

5. Core Transformations

Transformation	Use it for	Watch out for
Remove / Choose columns	Reduce width; keep only model fields.	Removing by name is safer than positional logic if file order changes.
Keep / Remove rows	Filter headers, totals, blanks, date windows, test rows.	Filtering by text after type conversion can break if errors exist.
Use First Row as Headers	Promote file headers.	Combined folders often need header handling inside the transform sample file.
Split column	Separate codes, names, paths, delimited fields.	Choose delimiter occurrence carefully; right-most delimiter is useful for file extensions.
Trim / Clean	Remove spaces and non-printing characters.	Clean does not fix all whitespace; sometimes replace non-breaking space manually.
Replace values	Standardize labels, nulls, bad codes.	Use exact match or conditional columns when partial text could over-replace.
Fill down / up	Normalize reports with grouped headings.	Fill before removing grouping rows; validate row counts afterward.
Merge	Join tables using matching keys.	Key columns must have compatible data types; check duplicates before expecting one-to-one joins.
Append	Stack similar tables or files.	Append matches by column name, not column position; missing columns become null.
Group By	Aggregate rows; create nested tables.	Use All Rows for advanced patterns, but it can be memory-heavy.
Pivot	Turn values into columns.	Needs unique row/column intersections or an aggregation.
Unpivot	Turn report-style month/category columns into rows.	Prefer Unpivot Other Columns when new period columns will appear later.
Conditional column	Create business categories.	For complex logic, custom M is often clearer.
Extract	Get text before/after delimiter, length, range.	Be explicit about delimiter direction and missing delimiter behavior.

Merge join types

- **Left outer:** keep all rows from left, match right. Default lookup behavior.
- **Right outer:** keep all rows from right, match left.
- **Full outer:** keep all rows from both sides.
- **Inner:** keep only matches.
- **Left anti:** rows in left with no match in right. Great for reconciliation.
- **Right anti:** rows in right with no match in left.
- **Fuzzy:** text similarity matching; useful, but audit the results.

Unpivot pattern

Use for cross-tab exports where months, metrics, or categories are columns.

1. Keep identifier columns selected, such as Customer, Product, Region.
2. Transform > Unpivot Columns > Unpivot Other Columns.
3. Rename `Attribute` to Period or Metric.
4. Rename `Value` to Amount, Quantity, or the real measure.
5. Change types and filter nulls.

Data types

Data type controls available transformations and how values are interpreted. Use **Change Type Using Locale** for ambiguous dates or numbers.

Null vs blank

`null` means missing value. Empty text `""` is still text. Replace blanks with null before fill, grouping, or validation.

Errors

Errors are values. You can keep, remove, replace, or inspect them. Do not hide errors until you know why they occurred.

6. M Language Essentials

Basic structure

```
let
    Source = Excel.CurrentWorkbook()
{[Name="Sales"]}[Content],
    #"Changed Type" = Table.TransformColumnTypes(
        Source,
        {{"OrderDate", type date}, {"Amount",
Currency.Type}}
    )
in
    #"Changed Type"
```

Rules to remember

- M is functional and case-sensitive: `Text.Trim` works; `text.trim` does not.
- Each `let` name points to a value. The `in` line returns the final value.
- Step names with spaces need `#"` and `"`, such as `#"Filtered Rows"`.
- Lists use `{}`, records use `[]`, tables use table functions.
- `each` is shorthand for a function with current row/element context.

Thing	Example	Meaning
List	`{1, 2, 3}`	Ordered collection. Access item with zero-based index: `MyList{0}`.
Record	`[Name="A", Qty=2]`	Named fields. Access field: `[Name]` or `Record.Field(row, "Name")`.
Table	`#table({"A"}, {{1}, {2}})`	Columns and rows. Most UI steps use `Table.*` functions.
Function	`(x as number) => x * 2`	Reusable expression with parameters.
Current row	`each [Amount] > 0`	Equivalent to `(_) => _[Amount] > 0`.
Optional access	`[Attributes]?[Hidden]? <> true`	Avoids errors when a field is missing.

7. M Snippet Library

Filter rows

```
Table.SelectRows(Source, each [Amount] > 0 and
[Status] = "Closed")
```

Select and rename columns

```
Table.SelectColumns(Source, {"OrderID",
"OrderDate", "Amount"})
Table.RenameColumns(Source, {{"Amt", "Amount"}})
```

Change types with culture

```
Table.TransformColumnTypes(
    Source,
    {{"InvoiceDate", type date}, {"Amount", type
number}},
    "en-GB"
)
```

Replace nulls or blanks

```
Table.ReplaceValue(Source, "", null,
Replacer.ReplaceValue, {"Customer"})
Table.ReplaceValue(Source, null, 0,
Replacer.ReplaceValue, {"Amount"})
```

Add custom column

```
Table.AddColumn(
    Source,
    "MarginPct",
    each if [Revenue] = 0 then null else [Profit] /
[Revenue],
    Percentage.Type
)
```

Group and aggregate

```
Table.Group(
    Source,
    {"Region", "Product"},
    {{"Revenue", each List.Sum([Revenue]),
Currency.Type}}
)
```

Merge lookup table

```
Merged = Table.NestedJoin(
    Sales, {"CustomerID"},
    Customers, {"CustomerID"},
    "Customer", JoinKind.LeftOuter
),
Expanded = Table.ExpandTableColumn(Merged,
"Customer", {"Segment"})
```

Create reusable function

```
(FileBinary as binary) as table =>
let
    Source = Excel.Workbook(FileBinary, true),
    Data = Source{[Item="Sheet1", Kind="Sheet"]}
[Data]
in
    Data
```

8. Excel vs Power BI Differences

Area	Excel	Power BI
Common destination	Worksheet table, PivotTable, connection only, or Data Model.	Semantic model table, load-disabled staging query, dataflow, or DirectQuery/composite model input.
Refresh	Manual refresh, refresh all, workbook connection properties, VBA/object model options.	Desktop refresh, scheduled service refresh, gateway credentials, incremental refresh policies.
Modeling after load	Worksheet formulas, PivotTables, Power Pivot relationships and measures.	Relationships, DAX measures, calculated tables/columns, report visuals, RLS.
Data volume	Worksheets have row limits; Data Model handles larger data but workbook size matters.	Designed for larger semantic models; still remove unneeded columns and rows early.
Staging	Use connection-only queries to avoid cluttering sheets.	Right-click query > Enable load off for helper queries.
Sharing	Queries live in the workbook and depend on accessible file paths/credentials.	Published datasets need credentials, gateway mapping, and service-compatible sources.

Excel loading choices

- **Table:** best when users need visible rows on a worksheet.
- **PivotTable:** quick analysis without creating a separate table output.
- **Only Create Connection:** best for staging and reusable query parts.
- **Add to Data Model:** use when relationships, measures, or larger-than-sheet data are needed.

Power BI loading choices

- **Import:** fastest visuals; data is loaded into the model.
- **DirectQuery:** queries source at interaction time; folding and source performance are critical.
- **Composite:** mix modes carefully; understand relationship and aggregation behavior.
- **Dataflows:** centralize reusable Power Query logic in the service.

9. Refresh and Deployment Checklist

- Confirm every source path is stable: SharePoint/OneDrive URL beats personal local path for shared work.
- Check credentials and privacy levels before handing off a workbook or publishing a dataset.
- Disable load for all staging/helper queries.
- Validate row count, distinct key count, null count, and error count after refresh.
- Document expected source schema and required columns.
- Use parameters for file paths, environment names, date ranges, or API endpoints.
- In Power BI, test refresh in the service, not only Desktop.
- For scheduled refresh, verify gateway, data source credentials, and OAuth token expiration.
- For incremental refresh, confirm Date/Time filter columns and folding before publishing.
- Keep sample files for folder-combine transformations representative of real files.

10. Performance and Query Folding

> **Query folding** means Power Query can translate steps into a source query, such as SQL, so the source system does more work and Power Query imports less data.

Fold-friendly habits

- Filter rows early.
- Remove unused columns early.
- Use source-native views for complex logic.
- Keep data type conversions simple.
- Check View Native Query after major steps.

Common folding breakers

- Custom row functions.
- Index columns.
- Sorting at the wrong time.
- Complex text parsing.
- Combining incompatible sources.

DirectQuery rule

For DirectQuery or Dual storage mode tables, folding is not a nice-to-have. It is central to whether the design is viable.

Problem	Better pattern	Reason
Huge refresh from SQL table	Filter date and select columns before merges.	Less data crosses the wire; more folding likely.
Power Query doing heavy calculations	Push logic into SQL view, warehouse, lakehouse, or dataflow when shared.	Centralized compute is easier to refresh and govern.
Slow append of many files	Filter folder list before invoking transform function; remove hidden/temp files.	Avoid opening files you will discard later.
Repeated expensive query	Reference staging query once; consider buffering only after testing.	Reduces duplicate logic; buffering can help or hurt depending on source and folding.
Native SQL step blocks later folding	Put as much logic as possible inside the native query, or use a view.	Later steps may not fold after a native query.

11. Buffering: Use With Care

When it can help

- Small lookup list/table reused many times.
- Preventing repeated enumeration of a volatile or expensive non-folding source.
- Stabilizing a list before membership checks such as `List.Contains`.

When it can hurt

- Large tables that should fold to the source.
- Steps before filters that would otherwise reduce rows.
- Refreshes already constrained by memory.

```
BufferedCustomers = Table.Buffer(Customers)
BufferedIDs = List.Buffer(CustomerIDList)
```

12. Parameters, Functions, and Reuse

Parameters are best for

- File or folder path.
- Environment: Dev/Test/Prod.
- Date windows for refresh or testing.
- Server, database, schema, or endpoint names.
- User-controlled thresholds such as minimum margin.

Functions are best for

- Transforming each file in a folder.
- Calling one API endpoint repeatedly.
- Applying the same clean-up logic to multiple tables.
- Parsing a standard text pattern.
- Creating custom date, fiscal period, or key logic.

Function template with clear parameters

```
(StartDate as date, EndDate as date, Region as text) as table =>
let
    Source = Sales,
    Filtered = Table.SelectRows(
        Source,
        each [OrderDate] >= StartDate
            and [OrderDate] <= EndDate
            and [Region] = Region
    )
in
    Filtered
```

13. Folder Combine: The Safer Pattern

1. Connect to the folder, not to one file.
2. Filter hidden, temporary, archived, and wrong-extension files before reading binaries.
3. Use the Combine Files experience, then inspect the sample transform query.
4. Move file-specific cleanup into the transform function.
5. Keep file metadata columns such as Name, Date modified, or Folder Path when useful for auditing.
6. Use a representative sample file. If the sample file is unusual, your function will be unusual too.

```
VisibleXlsx =
    Table.SelectRows(
        Folder.Files(p_FolderPath),
        each [Extension] = ".xlsx"
            and [Attributes]?[Hidden]? <> true
            and not Text.StartsWith([Name], "~$")
    )
```

Schema drift

Use `MissingField.UseNull` when columns may be missing and you want refresh to continue.

Header rows

Remove report titles and notes before promoting headers; otherwise column names will drift.

Auditability

Keep source file name in final data when finance or operations teams need traceability.

14. Data Quality and Validation

Check	How	Why it matters
Row count	`Table.RowCount(Source)` or Group By count.	Catches missing files, filter mistakes, and duplicate appends.
Duplicate keys	Group By key columns, count rows, filter count > 1.	Prevents unexpected row multiplication in merges.
Null required fields	Filter null in key/date/amount columns.	Protects relationships and calculations.
Invalid dates	Change type using locale; keep errors.	Date parsing issues are common in international files.
Category drift	Reference allowed values table, use anti-join.	Finds new statuses, regions, product codes, or misspellings.
Reconciliation	Compare source totals to final totals.	Confirms transformations did not lose or duplicate value.

Try...otherwise for controlled parsing

```
Table.AddColumn(
    Source,
    "ParsedDate",
    each try Date.FromText([RawDate], "en-GB")
otherwise null,
    type date
)
```

Find rows not in a reference table

```
Table.NestedJoin(
    Sales, {"ProductCode"},
    Products, {"ProductCode"},
    "Match",
    JoinKind.LeftAnti
)
```

15. Privacy, Credentials, and Security

Privacy levels

- **Public:** safe to combine broadly.
- **Organizational:** internal data.
- **Private:** sensitive data; most restrictive.

Formula.Firewall

Usually means Power Query is protecting data from leaking between sources. Fix source privacy levels, stage queries, or redesign the combine path.

Native queries

Use least-privilege read-only accounts. Native SQL can execute more than simple reads depending on permissions and statement.

> **Shared-workbook habit:** never hard-code personal desktop paths into production queries. Parameterize paths or use SharePoint/OneDrive URLs that collaborators can access.

16. Power BI Bonus: Model-Ready Query Design

Build star schema outputs

- **Facts:** numeric events at a clear grain, such as one row per invoice line.
- **Dimensions:** descriptive lookup tables: Date, Customer, Product, Region.
- **Keys:** stable, typed, unique on dimension side.
- **Measures:** write aggregations in DAX, not Power Query.

Incremental refresh essentials

- Create `RangeStart` and `RangeEnd` parameters as Date/Time.
- Filter the table date column with `>= RangeStart` and `< RangeEnd`.
- Verify the filter folds for relational sources.
- Configure policy in Power BI Desktop, publish, then let the service create partitions.
- Use an unchanging Date/Time field for refresh windows.

```
FilteredRows =
    Table.SelectRows(
        Source,
        each [OrderDateTime] >= RangeStart
            and [OrderDateTime] < RangeEnd
    )
```

Design decision	Power Query	DAX / model
Remove unused columns	Yes	No need to load them first.
Fix source data types	Yes	Model can format, but should receive correct types.
Create Date table	Often DAX/model, sometimes Power Query	Use one authoritative calendar table.
Revenue, margin, count measures	No, unless row-level source field	Use DAX measures for filter-aware analytics.
Business mapping table	Yes, as a dimension or merge input	Relate instead of flattening when reusable.

17. Excel Bonus: Workbook-Friendly Query Design

Good worksheet outputs

- Stable column names and order.
- No temporary helper columns unless users need them.
- Friendly text labels and formatted date/number types.
- One table per output purpose.

Good Power Pivot inputs

- Connection-only staging queries.
- Clean dimensions and facts loaded to Data Model.
- No merged report-style wide tables when relationships are better.
- Validate keys before loading to model.

18. Troubleshooting Playbook

Symptom	Likely cause	Fix
Expression.Error: column not found	Source schema changed or earlier rename removed it.	Inspect the first failing step; use `MissingField.UseNull` if missing columns are allowed.
Date conversion errors	Locale mismatch or invalid text.	Use Change Type Using Locale; add validation columns for day/month/year.
Merge creates too many rows	Right table key has duplicates.	Group right table by key and count; remove duplicates only when business-safe.
Refresh works locally, fails in service	Credentials, gateway, unsupported local path, OAuth, or privacy levels.	Configure data source credentials and gateway; replace local files with SharePoint/OneDrive or accessible network paths.
Formula.Firewall	Combining sources with privacy restrictions.	Set appropriate privacy levels, stage sources, or restructure so private data is not passed into public queries.
Very slow refresh	No folding, too many columns/rows, expensive custom functions, file combine reading everything.	Reduce early, check native query, push logic to source, filter file list before reading binaries.
Totals changed after cleanup	Filtered rows, duplicate joins, type conversion to null/error, or append mismatch.	Build reconciliation checks after each major layer.
Cannot edit source credentials	Credential cache or source path changed.	Data source settings > clear permissions, then reconnect with correct auth.

19. Final Review Checklist

- Every final table has a clear business purpose and grain.
- Every loaded table has only required columns.
- Data types are explicit and correct.
- Dates parsed with correct locale.
- Staging/helper queries are not loaded.
- Merge keys are typed the same and checked for duplicates.
- Row counts and totals reconcile to source.
- Refresh succeeds after closing and reopening the file.
- Shared paths and credentials work for the intended owner.
- Important business rules are named or commented.
- Query folding has been checked for relational sources.
- Power BI service refresh has been tested after publish.

20. Sources and Further Reading

Primary references used for this cheat sheet. Microsoft Learn content is updated over time, so use these links for product-specific edge cases.

- What is Power Query? https://learn.microsoft.com/en-us/power-query/power-query-what-is-power-query
- Power Query M formula language reference: https://learn.microsoft.com/en-us/powerquery-m/
- Query folding guidance in Power BI Desktop: https://learn.microsoft.com/en-us/power-bi/guidance/power-query-folding
- Using parameters in Power Query: https://learn.microsoft.com/en-us/power-query/power-query-query-parameters
- Data types in Power Query: https://learn.microsoft.com/en-us/power-query/data-types
- Merge queries overview: https://learn.microsoft.com/en-us/power-query/merge-queries-overview
- Append queries: https://learn.microsoft.com/en-us/power-query/append-queries
- Configure incremental refresh for Power BI semantic models: https://learn.microsoft.com/en-us/power-bi/connect-data/incremental-refresh-configure
- Data Privacy Firewall: https://learn.microsoft.com/en-us/power-query/data-privacy-firewall