STRETCHLEARN · WORKBOOK

Excel Advanced Formulas — Workbook

This workbook turns the course into formulas you actually build on real data. Work through one section per module: replace fragile VLOOKUPs with XLOOKUP and INDEX/MATCH, turn business rules into IF, IFS, and error-handled logic, run weighted and multi-criteria math with SUMPRODUCT, then build self-updating reports with FILTER, SORT, UNIQUE, and SEQUENCE. Use the worksheets to plan each formula, do every build step in your own Excel, and reach for the three editable templates as starting files. By the end you will have a set of robust lookups, conditional formulas, multi-criteria calculations, and a live dashboard you can reuse.

Lookups That Do Not Break: XLOOKUP and INDEX/MATCH

Audit your fragile lookups, rebuild them with XLOOKUP and INDEX/MATCH, and add two-way lookups that survive column changes.

Exercise: Break a VLOOKUP on Purpose

Find one VLOOKUP in your work that references a return column by number, then prove how fragile it is and replace it. Do this in your real workbook so the lesson sticks.

• Write down the VLOOKUP formula and the column number it uses as its third argument.

• Insert one column inside the lookup table, then record what the VLOOKUP now returns and why it is wrong.

• Rewrite it as XLOOKUP(lookup_value, lookup_array, return_array) pointing at real ranges, and confirm it still returns the right value after the column insert.

• If you are on Excel 2019 or earlier, write the INDEX(return, MATCH(value, lookup, 0)) version instead and note it for the INDEX/MATCH worksheet.

Worksheet: XLOOKUP Builder

Plan each XLOOKUP before you type it by filling in every argument. Use this for an exact-match lookup, then again for an approximate-match tier table.

lookup_value (the cell holding what you search for)

lookup_array (the single column/row you search in)

return_array (the column/row you want a value from)

if_not_found (the message instead of #N/A, e.g. "Not found")

match_mode (0 exact, -1 next smaller, 1 next larger, 2 wildcard)

search_mode (1 top-down, -1 bottom-up for latest record)

Final formula as typed

Worksheet: INDEX/MATCH and Two-Way Lookup Planner

Recreate a lookup with INDEX and MATCH, then plan a two-way lookup on a grid such as price by product and region. Remember MATCH returns a position and INDEX returns the value at that position.

Return range for INDEX (column the answer comes from)

MATCH lookup_value (what you are finding)

MATCH lookup_array (column to search) and match type (0 = exact)

Two-way: row MATCH (label range)

Two-way: column MATCH (header range)

Two-way INDEX data block (the values in the middle)

Final INDEX/MATCH formula as typed

Checklist: Robust Lookup Checklist

- [ ] Replaced at least one column-number VLOOKUP with XLOOKUP or INDEX/MATCH
- [ ] Used an if_not_found message instead of leaving a raw #N/A
- [ ] Built one approximate-match (match_mode -1) lookup on a sorted threshold table
- [ ] Built one left-lookup that VLOOKUP could not do
- [ ] Built one two-way lookup using INDEX with two MATCH functions
- [ ] Inserted a column inside a source table and confirmed the new formulas still work
- [ ] Noted which colleagues are on Excel 2019 or earlier so shared files use INDEX/MATCH

Conditional Logic: IF, IFS, and Error Handling

Turn real business rules into IF, IFS, AND, and OR formulas, then trap the right errors with IFNA and IFERROR.

Worksheet: Business Rule to Formula

Write a tiered rule from your work in plain English first, then translate it. Build it as nested IF and as IFS so you can compare readability and keep the version your Excel supports.

The rule in plain English (e.g. 90+ is A, 80+ is B, 70+ is C, else F)

The cell that holds the value being tested

Thresholds in order from highest to lowest

Result for each threshold

Nested IF version as typed

IFS version with a TRUE catch-all as typed

Which version I kept and why

Exercise: Combine Conditions with AND and OR

Build formulas where more than one thing must be true, and where any of several things triggers an outcome. Use AND, OR, and NOT inside the logical_test of IF on your own data.

• Write one IF(AND(...)) rule where two conditions must both be true, e.g. spend over a threshold AND a member.

• Write one IF(OR(...)) rule where any condition triggers a flag, e.g. overdue OR over a credit limit.

• Write one rule that combines AND with OR, e.g. spend enough AND (member OR VIP).

• Express one AND rule as deep nested IF instead, and note why the AND version is easier to change later.

Worksheet: Error-Handling Decision Sheet

For each formula that can error, decide whether to trap it and with which function. The rule of thumb: IFNA for lookups, XLOOKUP's if_not_found where possible, IFERROR only for genuine divide-by-zero.

Formula that can produce an error

Which error it can throw (#N/A, #REF!, #VALUE!, #DIV/0!, #SPILL!)

Is the error expected (a real no-match) or a sign of a real problem?

Chosen handler (XLOOKUP if_not_found / IFNA / IFERROR / none)

Fallback value or message

Final formula as typed

Checklist: Logic and Errors Checklist

[ ] Wrote one tiered rule as both nested IF and IFS with a TRUE catch-all
[ ] Built one AND rule, one OR rule, and one combined AND/OR rule
[ ] Reviewed each IFERROR and asked whether a #REF! or #VALUE! should really be hidden
[ ] Switched at least one lookup wrapper from IFERROR to IFNA or XLOOKUP if_not_found
[ ] Used IFERROR only on a genuine divide-by-zero such as a ratio with a possibly empty denominator
[ ] Can name what each error value (#N/A, #REF!, #VALUE!, #DIV/0!, #NAME?, #SPILL!) means
[ ] Confirmed no complex formula is wrapped in IFERROR just to hide red errors

SUMPRODUCT and Multi-Criteria Math

Build line-item totals and weighted averages with SUMPRODUCT, then count and sum with multiple criteria and pick the right tool.

Exercise: Weighted Average and Line-Item Total

Use SUMPRODUCT as multiply-down-then-add-up on your own numbers. Build a total and a weighted average, and verify each against a manual helper-column version.

• Build SUMPRODUCT(quantity, price) for a line-item total and confirm it matches a helper column of row-by-row products summed.

• Build a weighted average SUMPRODUCT(scores, weights); if your weights do not sum to 1, divide by SUM(weights).

• Introduce a blank or text cell into one array and observe how it is treated as zero — what row silently dropped out?

• Make the two arrays different sizes on purpose and record the error you get (#VALUE!).

Worksheet: Multi-Criteria Formula Planner

Plan a count and a sum with two or more conditions. Remember: multiply the condition arrays for AND, add them for OR. Build the same thing with COUNTIFS/SUMIFS where possible to compare.

Question in plain English (e.g. West orders over 1000)

Condition 1 (range and test)

Condition 2 (range and test)

AND or OR between them (multiply vs add)

Value range to sum (leave blank for a count)

SUMPRODUCT version as typed

COUNTIFS / SUMIFS equivalent as typed

Worksheet: Tool Selection: SUMPRODUCT vs SUMIFS vs Helper Column

For one real calculation, choose the best tool using the course decision guide, balancing power, readability, and speed. Note any whole-column ranges you should narrow.

The calculation and its conditions

Does it need weighting, OR across columns, or math in the criteria? (yes = SUMPRODUCT)

Will others maintain it or is the logic complex? (yes = helper column)

Is the workbook large/slow? (yes = helper column or narrowed ranges)

Chosen tool and one-line reason

Any whole-column references (e.g. B:B) I narrowed to the used range

Final formula as typed

Checklist: Multi-Criteria Math Checklist

[ ] Built a line-item total with SUMPRODUCT and verified it against a helper column
[ ] Built a weighted average, dividing by SUM of weights if they did not total 1
[ ] Counted with two conditions using both COUNTIFS and SUMPRODUCT and matched the results
[ ] Built one SUMPRODUCT that COUNTIFS could not do easily (OR across columns)
[ ] Used the double-unary minus to coerce a single condition to 1/0 where needed
[ ] Chose SUMIFS, SUMPRODUCT, or a helper column deliberately for one real calculation
[ ] Narrowed any whole-column array ranges to the actual used rows for speed

Dynamic Arrays: FILTER, SORT, UNIQUE, and Live Reports

Build self-updating reports with FILTER, SORT, SORTBY, and UNIQUE, generate series with SEQUENCE, and fix #SPILL!.

Exercise: Build an Interactive FILTER Report

Create a FILTER that spills matching rows and updates from a criterion cell. Requires Microsoft 365 or Excel 2021 or later. Add conditions and an if_empty message.

• Build FILTER(array, include, "No matches") for a single condition and confirm it spills with the blue outline.

• Point the condition at a criterion cell (e.g. a region in G1) and change G1 to watch the table rebuild itself.

• Add a second condition using multiply for AND, then change it to add for OR, and note the different results.

• Nest it as SORT(FILTER(...)) so the filtered rows come out already ordered.

Worksheet: Self-Updating Summary Planner

Design a summary that maintains itself using UNIQUE, SORT, and a spilled-range reference. The hash operator (e.g. F2#) lets one dynamic array feed another.

Category column to deduplicate (for UNIQUE)

SORT direction for the distinct list (1 ascending, -1 descending)

Distinct sorted list formula, e.g. SORT(UNIQUE(B2:B100))

Cell where that list spills (to reference with #)

Total formula beside it, e.g. SUMIF(range, F2#, values)

Top-N idea, e.g. SORT(FILTER(...), n, -1)

Live distinct-count, e.g. ROWS(UNIQUE(...))

Exercise: SEQUENCE and Fixing #SPILL!

Generate a series with SEQUENCE, reuse a spilled range with the hash operator, then create and resolve a #SPILL! error so you recognise it instantly.

• Use SEQUENCE to build a threshold column (e.g. SEQUENCE(10,1,0,100)) or a 30-day date axis (SEQUENCE(30,1,TODAY())).

• Reference one earlier spilled range with the hash operator in a COUNTA or a chart so it resizes with the data.

• Place a value in the path of a spilling formula to trigger #SPILL!, then read the dotted blocked-range outline.

• Clear the obstruction and confirm the array spills with no change to the formula itself.

Checklist: Dynamic Arrays Checklist

[ ] Built a FILTER with an if_empty message that spills matching rows
[ ] Made a FILTER interactive by pointing its condition at a criterion cell
[ ] Added AND (multiply) and OR (add) conditions inside FILTER
[ ] Built a sorted distinct list with SORT(UNIQUE(...))
[ ] Used the hash spilled-range reference (e.g. F2#) to feed a SUMIF or chart
[ ] Generated a number or date series with SEQUENCE
[ ] Triggered #SPILL!, read the blocked range, and fixed it by clearing the obstruction

Your Action Plan

1. Audit your workbooks for VLOOKUPs that reference a return column by number and rewrite the riskiest as XLOOKUP or INDEX/MATCH.
2. Add if_not_found messages (XLOOKUP) or IFNA wrappers to lookups that legitimately might miss, so #N/A stops appearing in reports.
3. Build one two-way lookup with INDEX and two MATCH functions on a grid such as price by product and region.
4. Convert one tiered rule to IFS with a TRUE catch-all, and combine multi-part rules with AND and OR inside IF.
5. Review every IFERROR and switch lookup wrappers to IFNA, keeping IFERROR only for genuine divide-by-zero.
6. Replace a manual line-item total with SUMPRODUCT(quantity, price) and build one weighted average.
7. Rebuild a multi-criteria count or sum, choosing SUMIFS, SUMPRODUCT, or a helper column deliberately and narrowing any whole-column ranges.
8. Build an interactive FILTER report driven by a criterion cell, with an if_empty message.
9. Create a self-updating summary with SORT(UNIQUE(...)) plus a spilled-range SUMIF using the hash operator.
10. Use SEQUENCE to generate a threshold or date series, and practise recognising and clearing a #SPILL! obstruction.