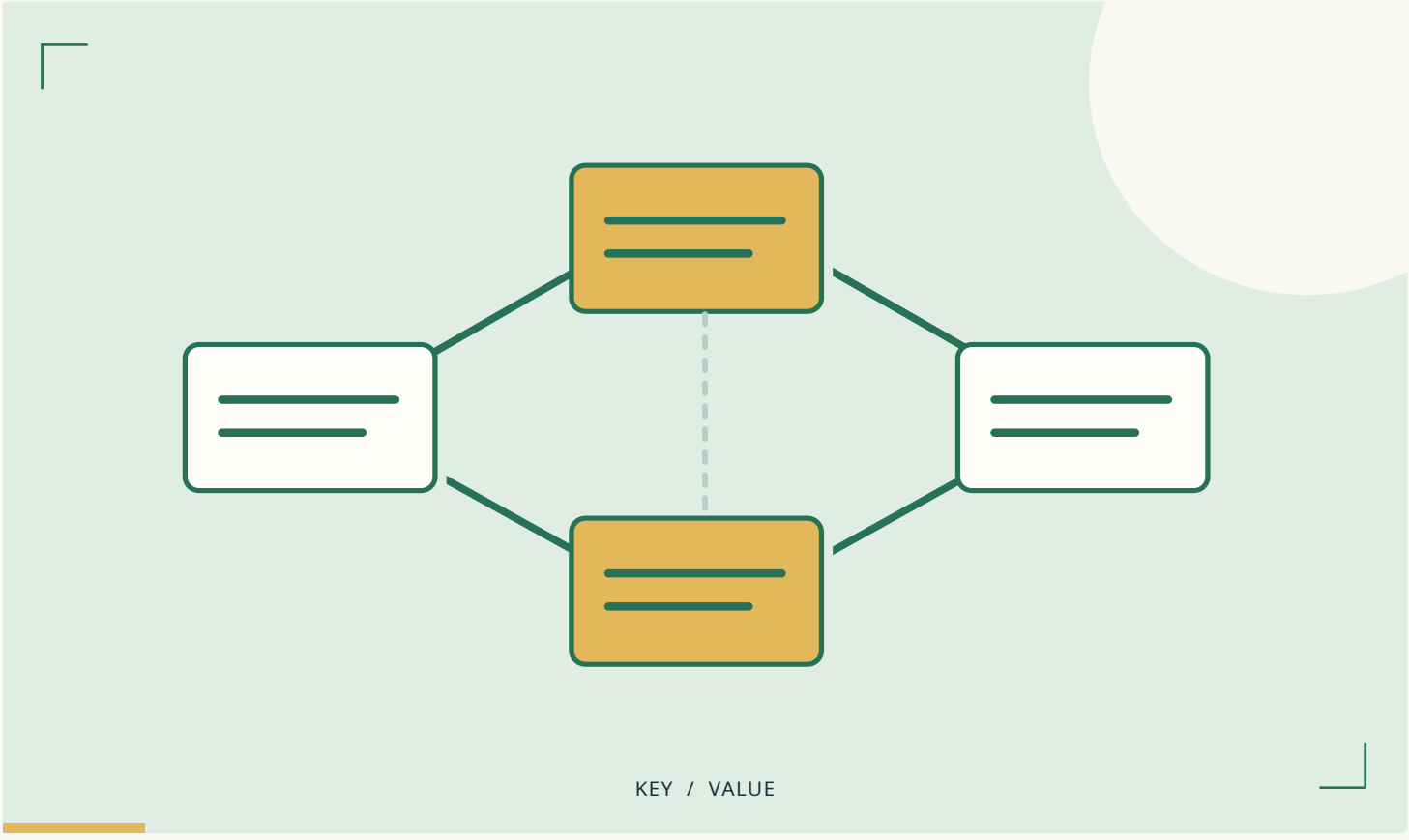
EXACT-MATCH REFERENCE LOOKUPS

# Retrieving Reference Values with Exact-Match Lookups




OFFICE & ADMINISTRATION

# Retrieving Reference Values with Exact-Match Lookups

## Learning outcomes

- Retrieve storage locations for ten registered document codes.
- Mark two unregistered codes instead of returning nearby values.
- Test uniqueness and identifier formatting before trusting a lookup.
- Build exact key-to-attribute mappings with explicit input and reference ranges.
- Distinguish missing input, unregistered keys, duplicate reference keys, and missing return fields.
- Preserve identifier strings and reject unsupported near-match corrections.
- Diagnose copy drift, array misalignment, stale boundaries, and source-record corruption.
- Verify lookup dependencies through complete-row permutation, attribute edits, and reference membership changes.

Audience: General workplace learners using synthetic practice materials

### Setup

Use only the supplied fictional records. Do not access real accounts, visit practice URLs, change live permissions, execute downloads, or submit messages. For spreadsheet exercises use a blank local practice workbook with English Excel formula names and comma separators; some regional installations use semicolons. Keep raw inputs and worked outputs separate.

Study the foundations, work through the applied units, complete the practice tasks, and then attempt the assessment. Use only the supplied fictional practice data. Keep your original inputs and your revised work separate.

## Core concepts

## Understand keys and return values

A lookup joins a requested code to a reference table and returns a related value, here a storage location. The code is the key, and the location is the return value. Keys should be stable identifiers, not a guess based on row position or a document’s display title. In this course codes are text such as DOC-04. Preserve the full code, including the prefix and leading zero. The reference table has ten unique keys; the input list deliberately includes two codes that are not registered.

## Use an explicit exact-match lookup

In modern Excel, =XLOOKUP(A2,$F$2:$F$11,$G$2:$G$11,"UNREGISTERED",0) searches for A2 in the reference codes and returns the corresponding location. The 0 states exact-match mode. XLOOKUP uses exact matching by default, but writing it explicitly documents the requirement. If the code is absent, the chosen text marker appears. An approximate match would be inappropriate: a nearby code can identify a different storage location. The reference table need not be sorted for this exact-match setup.

## Handle missing and duplicate keys separately

UNREGISTERED means the key was not found, not that the correct location is blank or that the file may be stored anywhere. Keep the input code visible so a reviewer can resolve it. XLOOKUP normally returns the first matching entry in its search order; if the reference has duplicate keys with different locations, an answer can appear without exposing the ambiguity. Check reference-key uniqueness separately. Do not “repair” an unknown code by trimming arbitrary characters or substituting a similar code without a defined normalization rule.

## Verify the join, not only the output count

Compare each returned location with its exact reference row. Test the first, last, and an absent code. In a temporary copy, reorder the reference table and confirm the results remain unchanged because the lookup uses keys rather than row positions. A legacy exact-match alternative is =IFNA(VLOOKUP(A2,$F$2:$G$11,2,FALSE),"UNREGISTERED") when available; its lookup key must be in the first column. Record the function used and preserve missing-code markers instead of hiding them as blanks.

## Worked example

A miniature reference contains DOC-21 → Shelf-A and DOC-23 → Shelf-C. Request DOC-22 is absent. An exact-match lookup returns UNREGISTERED, not Shelf-A because 21 is close to 22. Request DOC-23 returns Shelf-C even if the reference rows are reordered. If DOC-23 appears twice with different locations, the table needs resolution before its first-match result can be trusted. This illustrates three separate checks: exact key matching, absence handling, and reference uniqueness.

## Foundation practice

### Layout

Input codes in A2:A13, header A1 Code. B1 Location; write the lookup in B2 and fill to B13. Reference headers F1 Code and G1 Location; reference data in F2:G11. Use Excel Microsoft 365/2024/2021 for XLOOKUP, or the stated exact VLOOKUP alternative if supported.

### Reference rows

- DOC-01 -> Store-A1
- DOC-02 -> Store-A2

- DOC-03 -> Store-B1
- DOC-04 -> Store-B2
- DOC-05 -> Store-C1
- DOC-06 -> Store-C2
- DOC-07 -> Store-D1
- DOC-08 -> Store-D2
- DOC-09 -> Store-E1
- DOC-10 -> Store-E2

**Input codes**

- DOC-04
- DOC-01
- DOC-10
- DOC-03
- DOC-07
- DOC-02
- DOC-09
- DOC-05
- DOC-08
- DOC-06
- DOC-00
- DOC-11

**Rules**

Do not substitute approximate matches. Preserve codes as text. Unknown inputs must display UNREGISTERED. Reference keys are unique in the supplied dataset.

## Foundation tasks

- Build the twelve-row lookup output: ten registered codes plus two unregistered codes.
- Add a reference uniqueness check such as COUNTIF for each reference key and explain why any count above one would require review.
- Capstone: reverse the reference row order in a test copy and compare all twelve results with the baseline. Do not reorder the input list.

# 01 Establish the reference contract

An exact lookup answers a bounded question: which value is associated with this requested identifier in this particular reference snapshot? It does not decide whether an item exists anywhere, whether the reference is authoritative for every purpose, or whether two similar codes mean the same thing. The fictional Harbor Archive uses stable MAP codes to route requests to storage racks. A code identifies the record; a title describes it; a location and owner are attributes of that record.

Open assets/archive_requests.csv on a sheet named Archive at A1. Fourteen requests occupy A2:B15. Import assets/archive_reference.csv at H1 on the same sheet; ten reference records occupy H2:K11. The request headers are RequestID and RequestedCode. The reference headers are Code, Title, Location, and Owner. Create C1=Location, D1=KeyCount, E1=Decision, F1=Title, and G1=Owner, leaving the working output cells available for formulas. Do not overwrite either source table.

The supplied reference is deliberately unsorted. MAP-205 comes before MAP-107, for example. This tests whether the formula follows identifiers rather than physical positions. Every main reference code is unique, every main reference location is populated, and all keys use uppercase letters, digits, and a hyphen. These are properties to check, not reasons to stop checking future versions. The supplementary datasets deliberately break selected properties one at a time.

One row of the request table is one request, not one unique document. R04 and R11 both request MAP-107 and both need an answer. Repeated requests are legitimate here. Repeated reference keys create a different issue because the key is supposed to identify one reference record. Removing repeated requests to make the counts look unique would change the workload rather than repair the reference.

Use Microsoft 365, Excel 2021, or Excel 2024 for the XLOOKUP pathway. XLOOKUP is not available in Excel 2016 or Excel 2019; a later unit supplies a VLOOKUP and IFNA alternative. Examples use English function names and commas between arguments. Some installations use localized names or semicolons. Translate syntax for the application, not the dataset's identifiers. Save the working file as a workbook so formulas are retained.

The initial state must remain reproducible. Keep source CSVs unchanged and make controlled tests in a duplicate sheet or workbook. Record the reference version as Archive baseline and the range as H2:K11. A returned rack means that this snapshot maps the requested code to that rack. It does not grant permission to retrieve a real document or modify a real storage system. All materials in this course are fictional practice records.

### E01-X1 Establish the reference contract — practice

Build the Archive layout. Identify the two repeated request rows, the two nonblank codes absent from the reference, and the request with no code. State which field joins the tables and why RequestID is not that field.

Submit: A cell map and a five-row exception/repetition inventory.

# 02 Read the exact-match formula as a mapping

Start with one known request before filling a column. R01 has MAP-302 in B2. Find that exact text in H5, then move across the same reference row to J5, which contains Rack-F3. The output is not the fourth rack because this is the first request; it is the rack paired with the matching key. This manual trace establishes the mapping that the formula must reproduce.

The formula in C2 has five arguments. B2 is the lookup value from the current request. H2:H11 is the lookup array. J2:J11 is the return array. UNREGISTERED is the deliberate no-match marker. The final 0 selects exact match. Dollar signs keep the two reference arrays fixed as the formula is filled; B2 remains relative so each output reads its own request. The title column between H and J is not part of either array and causes no difficulty for XLOOKUP.

XLOOKUP uses exact matching by default, but writing 0 makes the requirement visible to a reviewer. In this pathway the optional search mode is omitted, which means ordinary first-to-last searching. Do not add binary search modes just because they sound faster: those modes impose sorting requirements that this unsorted table does not satisfy. A small, transparent reference is better served by the stated straightforward search.

For a second trace, B3 contains MAP-101. Its matching key is H2 and its location is J2, so C3 must read Rack-A2. For a third trace, MAP-330 is H11 and returns J11, Rack-G1. These examples deliberately span an interior row and both ends of the reference. Checking only adjacent requests could miss an incorrectly short range or a formula that accidentally follows row order.

The return array must stay aligned with the lookup array. Position one of H2:H11 belongs with position one of J2:J11, not with J3. Equal lengths are necessary but do not prove that the arrays describe the same records. If you move one array down a row, a formula can still produce plausible-looking racks while connecting the wrong records. Test meaning as well as whether Excel displays an error.

During this first trace, leave the blank-input distinction for the next unit. The raw formula is useful for understanding exact lookup, but the final report needs separate decisions for missing input and an absent nonblank key. A known match, a missing input, and an unregistered code are three states, not three ways to describe the same failure.

```
=XLOOKUP(B2,$H$2:$H$11,$J$2:$J$11,"UNREGISTERED",0)
```

### E02-X1 Read the exact-match formula as a mapping — practice

Enter the seed in C2, inspect each argument, and fill only C2:C4. Trace each result to a specific reference key cell and location cell. Explain why the title column does not need to be adjacent to the input.

Submit: Three outputs with source-cell evidence and an argument-by-argument explanation.

# 03 Separate missing input from an unregistered key

An empty RequestedCode is not a code that has been researched and found absent. R14 has no usable key. R12 and R13 have explicit keys, MAP-108 and MAP-999, but neither exists in the supplied reference. The report should preserve that difference because the next action differs: obtain a missing identifier versus verify or register a supplied identifier through the fictional reference process.

Wrap the exact lookup in an IF that first tests B2="". In this course source blanks are genuinely empty cells. The test also treats an empty text result as no supplied code, but it does not classify a cell containing a space as empty. Do not replace missing values with spaces to improve appearance. A space is text and can become a hidden identifier-quality problem rather than a reliable missing marker.

Use MISSING CODE for the first branch and UNREGISTERED only for a nonblank key with no reference match. These are visible text labels, not actual storage locations. Keep the input alongside them. If you replace both states with an empty output, a reviewer cannot distinguish an unprocessed row, a formula accidentally deleted, and a completed lookup with no match. A tidy-looking blank column is weaker evidence than a clear exception state.

Fill the guarded formula through C15. The ten distinct main keys appear across eleven requests because MAP-107 is requested twice. There should therefore be eleven retrieved rack values, two UNREGISTERED markers, and one MISSING CODE marker. These totals reconcile fourteen requests, but they do not establish that each rack is correct. Row-level comparisons are still required; swapping two racks preserves the totals while corrupting the report.

A missing or unregistered marker describes the current snapshot only. A new reference record can make a previously unregistered request resolvable. A corrected request can replace a typo. Neither change should be performed silently. Preserve the original request and identify the rule or source revision used for any correction so the report can be reproduced. This is especially important when a nearby code already exists and looks tempting.

Do not invent a default rack for exceptions. A default such as Rack-A2 would turn a failed match into a plausible but unsupported direction. The purpose of the marker is to stop that substitution. Likewise, zero is not a safe universal absence label: another lookup might legitimately return a numeric zero. Choose markers that are outside the domain of valid locations and explain their meanings.

```
=IF(B2="","MISSING CODE",
 XLOOKUP(B2,$H$2:$H$11,$J$2:$J$11,"UNREGISTERED",0))
```

### E03-X1 Separate missing input from an unregistered key — practice

Fill the guarded formula to C15. List all fourteen outputs in request order, reconcile the three result states, and explain why R12 and R14 must not share a label.

Submit: A completed location column and a state reconciliation.

# 04 Prove that copying preserves the intended search

A lookup formula has two different movement rules. The request reference must move down with the output row, while the reference arrays must remain fixed. In C2 the request is B2. In C3 it must be B3. In C15 it must be B15. The lookup array stays $H$2:$H$11 and the return array stays $J$2:$J$11 throughout. Inspect the formulas, not just the displayed values, because a plausible result can conceal a reference error.

Consider an incorrect seed with H2:H11 and J2:J11 left relative. Copying it down one row changes the searched window to H3:H12 and J3:J12. That removes MAP-101 at row 2. R02 then fails even though MAP-101 is registered. The error is caused by the moving boundary, not by a misspelled request. The first row may still look correct, so a successful seed is not proof of a successful fill.

Another mistake is locking $B$2 as the request. All copied formulas then search for MAP-302 and repeat Rack-F3. Reference arrays can be perfectly anchored while the report still fails because every row asks the same question. A third mistake locks only one endpoint, producing a shrinking or expanding search window. Write both endpoints explicitly and compare the seed, an interior formula, and the final formula before assessing the output.

Copying across is a separate operation. This course's location formula is designed for a downward fill. If you copy it into the Title column, B2 can shift to E2 and the return array remains J rather than switching to I. A correctly anchored formula does not infer your new reporting intention. Write the title formula deliberately with the same requested key and the correct title array instead of relying on horizontal copy behavior.

A useful diagnostic table has four fields: output cell, expected request cell, expected lookup range, and expected return range. Add the observed formula beside those fields. This makes the source of a wrong answer visible. If the arrays are correct and fixed, move on to identifier contents and reference quality. If the arrays drift, fix the formula first rather than editing source data until the result happens to look right.

When testing an intentionally broken seed, use a separate column or duplicate sheet. Do not overwrite your verified baseline. Restore the guarded formula after observing the failure. Controlled faults are valuable because they show which test detects which error, but an unmarked faulty formula left in a working report defeats the purpose of the exercise.

### E04-X1 Prove that copying preserves the intended search — practice

In a test copy, compare a relative-array formula and a locked-request formula against the baseline. State the exact formulas produced in row 3, predict the outputs for R02, then confirm the causes. Restore the baseline.

Submit: Two fault records with row-3 formulas, expected wrong outputs, causes, and restoration evidence.

# 05 Count reference matches before trusting a return

A lookup can return a value without proving that the key is unique. The main reference contract requires one row per code. Test that contract on the reference itself by placing L1=ReferenceCount and entering COUNTIF over H2:H11 for each H cell. Every supplied reference row should show one. A number above one identifies repeated reference keys, while a count of one says only that the key appears once inside the chosen range.

Use a separate request-side count in D2. For a missing request, return the visible label MISSING CODE; otherwise count the requested code in the fixed reference array. Fill D2:D15. Registered requests show one, the two unregistered requests show zero, and the blank request shows the missing marker. This diagnostic column supplies membership evidence independently from the returned location. It helps distinguish a no-match state from a matching row with a problematic return value.

COUNTIF is safe for the identifiers in these datasets because the code grammar excludes wildcard characters and comparison operators. It is not a universal literal-string uniqueness validator for every possible identifier format. If an organization's identifiers can include asterisks or question marks, the counting rule needs a deliberately escaped or otherwise literal comparison design. Do not generalize this classroom count beyond its stated key grammar.

Now create a decision column. Check missing input first, then zero matches, then more than one match, and finally the single-match case. The possible decisions are MISSING CODE, UNREGISTERED, AMBIGUOUS KEY, and READY. READY means the key count satisfies the one-reference-row rule; it does not certify every attribute's completeness or the physical accuracy of a rack label. Later units add a return-field quality check.

Repeated request rows do not make this count ambiguous. D5 and D12 are both one for MAP-107 because each formula searches the reference, not the request list. If you count B2:B15 instead, those rows show two, but that answers a different question: how many requests mention this code? A useful number can still be the wrong number for the decision being made. Label the range and purpose of each count.

Keep the reference boundary explicit. A duplicate hidden below row 11 will not be counted by H2:H11. A count of one is therefore evidence about the declared snapshot, not the entire worksheet or all future data. When a reference version grows, both lookup and diagnostic ranges must be reviewed together. An updated lookup with an outdated count gate can produce internally inconsistent decisions.

```
D2: =IF(B2="","MISSING CODE",COUNTIF($H$2:$H$11,B2))
L2: =COUNTIF($H$2:$H$11,H2)
E2: =IF(B2="","MISSING CODE",
 IF(D2=0,"UNREGISTERED",IF(D2>1,"AMBIGUOUS KEY","READY")))
```

### E05-X1 Count reference matches before trusting a return — practice

Fill request counts, reference counts, and decisions. Report D5, D12, D13, D14, D15, and all ten reference counts. Explain why counting the request list would not validate reference uniqueness.

Submit: A count audit with correctly distinguished request repetition and reference uniqueness.

# 06 Expose duplicate reference keys with a controlled case

Create a sheet named Duplicates. Import assets/duplicate_reference.csv at F1 so the four reference records occupy F2:G5. They contain MAP-101 → Rack-A2, MAP-107 → Rack-A7, MAP-220 → Rack-D2, and another MAP-107 → Rack-Z9. Enter A1=RequestedCode and A2=MAP-107. This case is not a request repetition: two reference records compete to define the location for the same key.

A standard first-to-last exact XLOOKUP returns Rack-A7. That answer is consistent with the function's search rule but insufficient for the reference contract. The existence of a returned value does not show that the mapping is unambiguous. Reversing the four complete reference rows makes Rack-Z9 appear first and changes the returned value even though the set of source rows has not changed. The ordering test reveals a conflict that the ordinary result hides.

COUNTIF over F2:F5 returns two for MAP-107. Use that number to block a final location with AMBIGUOUS KEY rather than choosing whichever rack happens to be first. Do not repair the ambiguity by using a last-to-first search. That simply selects the other competing row according to order; it does not provide a business rule for deciding which location is current. Search order is a technical mechanism, not a source-authority policy.

A duplicate can be problematic even if both rows currently contain the same rack. If the contract requires one record per key, identical duplicates still violate the contract and can diverge after later edits. Removing one may be an approved reference-maintenance action, but it should be separate from answering a request. In this exercise, record the issue and keep the original source evidence. Do not silently delete a row to make a count become one.

This distinction also explains why no-match handling is not duplicate handling. The if_not_found argument is used when the key is absent. A duplicate key is present, often more than once, so a no-match marker will not expose it. Adding IFNA around the formula does not make the reference unique. You need an explicit uniqueness check or another authoritative constraint before treating a returned location as reliable.

Restore the source order after the reversal test and keep both observations in your audit. The report should say that the first-match output changed from Rack-A7 to Rack-Z9 and that the match count remained two. These two observations together identify the problem: order determines the selected value because the key does not identify a single row. That is stronger than writing only that the lookup was wrong.

```
B2: =XLOOKUP(A2,$F$2:$F$5,$G$2:$G$5,"UNREGISTERED",0)
```

```
C2: =COUNTIF($F$2:$F$5,A2)
D2: =IF(C2=0,"UNREGISTERED",
 IF(C2>1,"AMBIGUOUS KEY",B2))
```

### E06-X1 Expose duplicate reference keys with a controlled case — practice

Run the duplicate case in original and reversed complete-row order. Record raw output, count, and gated decision in both states. Then explain why search mode -1 would not resolve the reference issue.

Submit: A two-state comparison and a non-destructive resolution note.

## 07 Distinguish key presence from return-field completeness

Create ReturnQuality and import assets/return_quality.csv at F1. RET-01 maps to Box-P1, RET-02 has a genuinely empty Location cell, and RET-03 maps to Box-P3. Enter requests RET-01, RET-02, RET-99, and a genuinely empty cell in A2:A5. RET-02 is registered but incomplete. RET-99 is not registered. These cases require different labels even if an unguarded output appears blank or zero.

A no-match marker is not a general missing-data detector. The key RET-02 is found in F3, so the if_not_found branch does not apply merely because G3 is empty. Do not rely on the appearance of a returned blank cell to classify this case; spreadsheet functions and cell formatting can make empty-source results confusing. Instead, inspect completeness in the reference and return a deliberately prepared status or display value.

Add H1=LocationState and I1=SafeLocation. In H2 use IF(G2="","MISSING LOCATION","PRESENT") and fill to H4. In I2 use IF(G2="","MISSING LOCATION",G2) and fill to I4. These helpers transform a known empty source field into an explicit marker before the lookup retrieves it. The empty-cell comparison also treats empty text as missing under this lesson's rule; a space is not accepted as a clean empty field.

Use a missing-input guard and an exact lookup from F2:F4 into I2:I4. The four outputs should distinguish a valid location, a registered key with no location, an absent key, and an absent input. Count the keys separately if uniqueness is not already established. A safe return helper does not solve duplicate keys, and a uniqueness gate does not populate a missing location. Each check has one specific purpose.

For this small case the state helper is transparent and auditable. An alternative compact formula might perform several lookups inside nested conditions, but a shorter expression is not automatically easier to verify. Prefer a design in which a reviewer can see the raw location, its quality classification, and the displayed lookup result. Keep the helper headers descriptive so MISSING LOCATION is not mistaken for an actual storage box.

If a later source revision fills RET-02 with Box-P2, the key remains registered and its quality state changes from missing to present. That is a return-field update, not a new key registration. If RET-99 is added, that is a separate reference-membership change. Recording the type of change helps you predict which tests should respond and avoids treating every exception as a spelling problem.

```
H2: =IF(G2="","MISSING LOCATION","PRESENT")
I2: =IF(G2="","MISSING LOCATION",G2)
B2: =IF(A2="","MISSING CODE",
 XLOOKUP(A2,$F$2:$F$4,$I$2:$I$4,"UNREGISTERED",0))
```

### E07-X1 Distinguish key presence from return-field completeness — practice

Build the two reference helpers and four guarded outputs. Temporarily fill G3 with Box-P2, record exactly which output changes, and restore the blank. Explain why UNREGISTERED was not correct for RET-02.

Submit: Four classified results and one controlled return-field update.

# 08 Preserve identifiers before matching them

Identifiers are labels, even when they contain only digits. Create TextKeys and import assets/identifier_text.csv at F1 with the Code column explicitly treated as text before import. The three keys are the four-character text 0042, the two-character text 42, and the five-character text A-042. They map to Bin-Zero, Bin-Short, and Bin-Prefix. The two digit-only keys are intentionally different identifiers in this fictional register.

CSV files do not carry Excel cell-type metadata. An automatic import can interpret 0042 as a number and discard its leading zeros. Formatting that already-converted cell as Text afterward does not recover the lost characters. A custom numeric display format can show zeros while the underlying value remains numeric. For identifiers, preserving the intended string before conversion is different from making a numeric value look similar on screen.

Enter A2 as text 0042, A3 as text 42, and A4 as text A-042. In direct cell entry, an initial apostrophe can tell Excel to store the following digits as text; the apostrophe is an entry device, not part of the code itself. Verify the stored string rather than copying a visible apostrophe into the source CSV. Exact lookups into the preserved reference should return the corresponding bins without padding, arithmetic, or numeric conversion.

Use EXACT as a separate string-comparison diagnostic. EXACT("0042","42") is FALSE because the characters differ. EXACT("MAP-107","map-107") is FALSE because EXACT is case-sensitive. Do not confuse that function's case rule with a claim that every lookup's exact mode enforces a case-sensitive identifier policy. In the main course dataset, keys are already uppercase canonical strings, so no case-variant matching assumption is needed.

There is no universal rule that every numeric-looking code should have four digits. If you pad 42 to 0042 in this register, you change a valid key into another valid key and retrieve the wrong bin. Normalization is safe only when a defined source contract says which representations are equivalent. A fixed-width policy in some other registry does not authorize changing this one. Preserve raw values and document any explicitly approved normalized field separately.

Microsoft also warns that long numeric values can lose precision when treated as numbers. A code is not made safer by being all digits. The broader lesson is to decide the data type from the meaning of the field before import, then test representative identifiers that differ in leading zeros, length, and prefixes. Matching cannot reconstruct distinctions that were already destroyed during conversion.

```
B2: =XLOOKUP(A2,$F$2:$F$4,$G$2:$G$4,"UNREGISTERED",0)
=EXACT("0042","42")
=EXACT("MAP-107","map-107")
```

### E08-X1 Preserve identifiers before matching them — practice

Retrieve the three text-key locations. Record the character count and type intention of each input, evaluate the two EXACT comparisons, and explain why padding text 42 would be incorrect here.

Submit: Three exact outputs and an identifier-preservation explanation.

## 09 Use the legacy exact-match pathway deliberately

The Archive case can also be solved with VLOOKUP and IFNA where those functions are available. VLOOKUP searches the first column of a rectangular table and returns a column selected by its position within that rectangle. In H2:K11, Code is column 1, Title is column 2, Location is column 3, and Owner is column 4. These indices are relative to the table, not the worksheet's absolute column numbers.

For the location output, use VLOOKUP(B2,$H$2:$K$11,3,FALSE). The final FALSE requests exact matching. Do not omit it. VLOOKUP's omitted or TRUE range_lookup argument uses approximate matching behavior and is unsuitable for these sparse identifiers. The reference does not need sorting for this FALSE pathway. A successful result on one known key does not prove that an approximate formula will handle an absent key safely.

Wrap the exact lookup in IFNA to show UNREGISTERED when no key is found. Add the same outer missing-input IF used in the XLOOKUP design. This preserves the distinction between a blank request and a supplied absent code. It also keeps the no-match treatment narrow: IFNA catches the #N/A error rather than replacing every possible formula error with a harmless-looking marker.

Suppose someone uses return index 5 with the four-column H:K table. That is a structural formula error because the requested column lies outside the selected table. Calling the result UNREGISTERED would blame the input code for a broken formula. A broad error wrapper can hide that distinction. Diagnose table dimensions and return indices instead of turning every error into a reference-membership claim.

VLOOKUP's first-column rule also matters when a layout changes. If Location is placed to the left of Code, a simple table starting with Code cannot return a column to its left using this form. XLOOKUP can use separate lookup and return arrays in either direction. Do not rearrange a reference silently just to make a copied formula work; identify the chosen method and preserve paired records if a deliberate layout transformation is needed.

Treat the two function pathways as alternative implementations of the same contract. They should agree for the supplied unique, complete, canonical keys and the same reference snapshot. If they disagree, compare the key cell, table boundaries, return index or array, exact-match setting, and exception wrapper before changing source data. Agreement is useful evidence, but both formulas can share the same wrong boundary, so retain manual row traces as well.

```
=IF(B2="","MISSING CODE",
 IFNA(VLOOKUP(B2,$H$2:$K$11,3,FALSE),"UNREGISTERED"))
```

### E09-X1 Use the legacy exact-match pathway deliberately — practice

Build a legacy location column in a separate area and compare all fourteen outputs with C2:C15. State the title and owner column indices. Diagnose a formula using index 5 without converting its structural error into UNREGISTERED.

Submit: A complete agreement check and a column-index diagnosis.

# 10 Retrieve several attributes without changing the key

A request may need more than a location. The Archive report also requires Title and Owner, but the identity rule does not change. F2 should look up B2 against H2:H11 and return I2:I11. G2 should use the same request and key array while returning K2:K11. Keeping the same key across attributes makes each output row a coherent description of one reference record rather than a collection of plausible values.

For R01, the matched reference row is H5:K5. The location is Rack-F3, the title is Equipment index, and the owner is Technical. For R02, the matching row is H2:K2, giving Rack-A2, Arrival checklist, and Intake. The title is not a replacement key: display titles can be revised or reused. A lookup on a descriptive field would need its own uniqueness contract, which is not supplied here.

Use the same missing and unregistered labels in each attribute column so the exceptions remain consistent. A missing requested code should not show MISSING CODE in Location but a borrowed title in Title. If one attribute uses a different request reference or a different key boundary, the report can accidentally combine records. Inspect the formulas for a known interior key, a repeated request, an absent key, and the missing input.

There is a useful difference between retrieving multiple attributes and concatenating them into one string. Separate columns preserve the fields for review and make a missing attribute visible. A combined display such as Rack-F3 | Technical can be convenient later, but it should not replace the underlying typed fields during verification. A separator or attractive format does not establish that the two values came from the same key.

For this lesson use one formula per attribute rather than a multi-column spill. That keeps the examples portable across the two chosen function pathways and makes each return range explicit. In the legacy pathway, change only the table-relative return index from 3 to 2 or 4, while preserving the key cell, table, FALSE, and exception logic. A new attribute should not require a new matching policy.

Check every main reference key at least once across the request list, not merely every owner group. Several records share an owner, so a wrong owner formula can coincidentally look correct on many rows. The repeated Technical values are a reminder that nonunique return fields provide weak diagnostic evidence by themselves. Combine them with the more specific location and title to verify the matched record.

```
F2: =IF(B2="","MISSING CODE",
```

```
 XLOOKUP(B2,$H$2:$H$11,$I$2:$I$11,"UNREGISTERED",0))
G2: =IF(B2="","MISSING CODE",
 XLOOKUP(B2,$H$2:$H$11,$K$2:$K$11,"UNREGISTERED",0))
```

### E10-X1 Retrieve several attributes without changing the key — practice

Fill Title and Owner for all requests. Report complete location/title/owner triples for R01, R04, R11, R12, and R14. Explain why two Technical results agreeing does not by itself prove the correct record was selected.

Submit: A multi-attribute report and five traced triples.

# 11 Diagnose equal-length but misaligned arrays

A formula can be syntactically valid and still implement the wrong mapping. In a duplicate Archive sheet, change only the return array from J2:J11 to J3:J12 while leaving H2:H11 fixed. Both arrays contain ten cells, so a simple length comparison will not reveal the fault. Position one now pairs MAP-101 with the location from the next reference row, Rack-C1, instead of Rack-A2.

Trace R01 under this fault. MAP-302 is the fourth key in H2:H11. The fourth cell of J3:J12 is J6, Rack-B4, so the formula returns a real rack belonging to MAP-118. The output is not an error string; it is a wrong association. This kind of defect is dangerous precisely because the value looks plausible and could pass a superficial check that every result begins with Rack-.

Trace R02 as a second test. Its key MAP-101 is at the first lookup position, but the shifted return array supplies J3, Rack-C1. R04's MAP-107 is at the third lookup position and incorrectly receives J5, Rack-F3. Together these traces show a systematic one-row offset. Do not correct the three results individually; repair the common return-array boundary and recheck the entire output column.

The final key MAP-330 reaches the final shifted return cell J12, which is outside the populated main reference. Do not make a broad assumption about how that empty return will display in every formula or format. The meaningful diagnosis is already available: the range pair no longer describes the same records. An output from an empty tail cell is additional evidence, not the sole test on which the repair depends.

A stronger alignment check records both starting and ending rows, confirms the same record ordering, and traces known keys to their source rows. Equal-sized arrays are one requirement, not the full requirement. Similarly, two arrays on different sheets can be aligned only if their order is governed by a common record contract. Matching their lengths after independent sorting does not restore the lost relationships.

After repairing the return array, compare every result with the baseline. Three spot checks found the defect, but they do not prove that all formulas were repaired. A partially filled correction could leave lower rows wrong. Inspect the first, middle, and last formulas and compare the full fourteen-row result list, including both absent keys and the missing-input guard. Keep the controlled fault separate from the final workbook.

### E11-X1 Diagnose equal-length but misaligned arrays — practice

Run the shifted-return fault in a copy. Give the wrong location and the source cell used for R01, R02, and R04. Repair the array and describe why a length-only check would have missed the defect.

Submit: Three causal traces and a complete-restoration check.

# 12 Reorder complete records, not isolated columns

For a unique-key reference, ordinary exact lookup should be invariant under a permutation of complete records. Reverse H2:K11 as whole rows in a test copy while leaving the request table untouched. Each code must travel with its title, location, and owner. The physical row of a code changes, but its associated attributes do not. The fourteen requested results should therefore remain exactly the same.

This is different from sorting only H2:H11. If you rearrange just the code column and leave I:K in place, you rewrite the mappings by attaching codes to other records' attributes. A lookup can faithfully return those now-corrupted pairings. The function has not failed to search; the reference has lost its record integrity. Do not describe such an experiment as proof that exact lookup requires a sorted table.

A small two-row example makes the distinction visible. Suppose Code A maps to Rack-1 and Code B maps to Rack-2. Reversing both complete rows leaves those associations unchanged. Reversing only the key column creates B → Rack-1 and A → Rack-2. The same exact lookup now returns different racks because the source facts were changed, not because the matching mode changed.

After a full-row reversal, trace MAP-101 and MAP-330 again. Their source rows swap ends, but their locations remain Rack-A2 and Rack-G1. A position-based formula such as returning the next cell in a fixed order would not have this property. The test distinguishes a key-based join from an accidental dependency on where a record happened to be located in the original source.

Reordering alone does not prove uniqueness. The earlier duplicate case changed its first-match result when row order changed, but duplicates with identical return values might not reveal themselves through output comparison. Always retain the explicit reference count audit. The two tests answer different questions: is each key unique, and do the formulas follow the mapping when complete records move?

Use a comparison log with original output, reordered output, and same/different for every RequestID. Do not reorder the requests to make comparison easier, because that removes the stable row-by-row basis for the test. A request identifier belongs to the request table and remains useful for joining test evidence back to the original workload. Restore the baseline source order after recording the results, or clearly label the reordered copy as a test version.

### E12-X1 Reorder complete records, not isolated columns — practice

Reverse all ten reference records as complete H:K rows in a copy. Compare every request output. Then explain the two-row isolated-column example without applying it to your final source.

Submit: A fourteen-row invariance log and an explanation of source-record integrity.

# 13 Reject near matches and invented corrections

Sparse identifiers are not measurement scales. MAP-107 and MAP-118 are registered, but that does not give MAP-108 a meaningful nearest neighbor. The numeric-looking suffix is part of a label, not a distance that authorizes substitution. The correct baseline result for MAP-108 is UNREGISTERED. A nearby registered code can point to another record and another rack, so approximate substitution can produce a confidently wrong route.

XLOOKUP match mode -1 allows an exact match or the next smaller item; mode 1 allows an exact match or the next larger item. Those modes serve different problems, such as explicitly defined bands, and are outside this identifier contract. Mode 2 enables wildcard matching. None is a general improvement on exact mode 0 for these codes. Do not change modes merely to reduce the visible number of exceptions.

A correction process starts with evidence. If a request has a leading or trailing space, a malformed prefix, or a transposed character, preserve the raw value and flag it for review. The fictional Archive contract supplies uppercase codes exactly as listed and does not authorize trimming, padding, case conversion, or nearest-code replacement. You may propose a correction in a separate note, but a proposal is not a verified key.

There is an important difference between detecting a suspicious string and repairing it. EXACT can show that two candidate strings differ. A length check can reveal an unexpected character count. Neither tells you which value the requester intended. If MAP-107 and MAP-170 were both valid, a transposed digit could lead to another legitimate record. Even an apparently obvious typo needs a defined source rule before it changes the lookup input.

Keep missing, unregistered, and ambiguous states separate during correction. A blank request needs a key supplied. An unregistered key needs its spelling or reference membership investigated. An ambiguous key needs the reference conflict resolved. Applying the same blanket transformation to all three can hide the underlying cause and make later reconciliation impossible. The decision labels are a workflow aid because they preserve these distinctions.

The practical deliverable should include an exception register with RequestID, raw RequestedCode, current decision, evidence, and proposed next step. In this exercise the next step is a written fictional action, not contacting a real person or editing a live registry. Record that R12 and R13 remain unresolved under the baseline. A reliable report can contain unresolved exceptions; reliability means representing the evidence accurately, not forcing every row to look successful.

### E13-X1 Reject near matches and invented corrections — practice

Create exception-register entries for R12, R13, and R14. Contrast exact mode 0 with modes -1, 1, and 2. Explain why choosing MAP-107 for MAP-108 would change the request rather than complete it.

Submit: Three exception entries and a matching-policy explanation.

# 14 Predict the effect of an attribute-only revision

A reference update can change an attribute without changing identity. In a copy of Archive, change MAP-107's Location from Rack-A7 to Rack-A9 while leaving its code, title, owner, and row position unchanged. There are two requests for that code, R04 and R11. Both should receive the revised location. No other request should change, and the reference uniqueness counts should remain one.

This test evaluates dependency, not only correctness at one snapshot. A hardcoded Rack-A7 in either output cell will fail to respond. A formula referencing a different key may change on the wrong row or not at all. Predicting the affected RequestIDs before changing the source makes the observation meaningful; otherwise a learner can mistake any visible update for success.

The change does not make a new code registered. The baseline still has ten unique reference keys, eleven registered requests, two unregistered requests, and one missing request. Title and Owner remain Visitor guide and Intake for both MAP-107 requests. If those fields change during a location-only test, inspect which source cells were edited and whether the report formulas use the intended return arrays.

A report refresh should describe the reference snapshot used. If someone compares yesterday's Rack-A7 output with today's Rack-A9 output without noting the location revision, they may incorrectly label one lookup as defective. A formula can be correct against each of two different snapshots while returning different values. Reproducibility requires both the formula and the source version, not just a screenshot of the final rack.

Restoration is part of the test. Put Rack-A7 back in the reference and verify that both request outputs revert. A successful forward change followed by a failed restoration can indicate a pasted value, a partially recalculated workbook, or an accidental edit to the wrong source cell. Keep the observed sequence explicit: baseline, revised attribute, restored baseline. Do not merge observations from separate copies without labeling them.

This controlled change also separates authoritative source edits from output edits. Changing C5 directly to Rack-A9 might make one row look correct, but it bypasses the mapping and leaves R11 stale. The reference contains the shared fact; the request outputs should derive from it. When the same identifier appears in many requests, that distinction is what makes lookup-based reporting maintainable rather than a series of manual substitutions.

### E14-X1 Predict the effect of an attribute-only revision — practice

Run the MAP-107 location-only change and restoration. Record the changed source cell, affected RequestIDs and output cells, unchanged title/owner values, and state totals in all three states.

Submit: A three-state dependency log with no unexplained edits.

# 15 Expand the reference boundary as a controlled change

Now test a membership revision rather than an attribute revision. In a copy of Archive, append assets/new_reference_row.csv as the new record H12:K12. The new key is MAP-108, with title Visitor supplement, location Rack-A8, and owner Intake. Do not overwrite MAP-107 or insert the row into the request list. This is a new reference record for a previously absent requested code.

Appending a source record does not automatically expand a fixed-range formula. If C13 still searches H2:H11 and returns J2:J11, R12 remains UNREGISTERED despite MAP-108 being visible in row 12. That result is correct for the old declared range but wrong for a report claiming to use the expanded reference. Distinguish a stale formula boundary from a failed exact-match algorithm.

Update all related boundaries together: lookup arrays H2:H12, location returns J2:J12, title returns I2:I12, owner returns K2:K12, request match counts H2:H12, and reference count checks through row 12. The legacy rectangle becomes H2:K12. A partial update is dangerous: a location can resolve while a stale count still says zero, or a title formula can keep reporting UNREGISTERED for the same request.

After the complete update, only R12 changes state. It becomes a single registered match with Rack-A8, Visitor supplement, and Intake. R13 remains UNREGISTERED and R14 remains MISSING CODE. The request count stays fourteen because the workload did not change. The reference count becomes eleven unique keys. Registered requests become twelve, with one unregistered and one missing request.

An append test should include both a known code and an absent code. If the only new reference key is never requested, all displayed request outputs may remain unchanged even after a correctly expanded boundary. That is not proof that the append was ignored; it means the test did not exercise the new mapping. Here MAP-108 was chosen deliberately because R12 supplies a visible boundary-sensitive case.

Restore the baseline by removing the appended test record from the copy and returning all formulas to the baseline ranges, or retain the expanded copy under an explicit new version label. Do not remove the row while leaving a report labeled as the expanded snapshot. Document both the old-range failure and the fully updated result. A robust audit explains not just the final answer but why the first append attempt did not change it.

### E15-X1 Expand the reference boundary as a controlled change — practice

Append the new reference row in a copy. Record R12 with old boundaries, then update every lookup and diagnostic boundary and record the new result and state totals. Specify the revised seed location and key-count formulas.

Submit: A boundary-change log, exact formulas, and a complete reconciliation.

# 16 Audit the original document-code foundation

Return to the original DOC practice supplied with the course. Its layout differs from Archive: input codes are A2:A13, outputs are B2:B13, and the ten Code/Location reference records are F2:G11. Do not transplant the Archive formula without changing its cell map. Reusing a method is appropriate; reusing unrelated addresses is not. Keep the original practice on its own sheet so both mappings remain independently reproducible.

The original reference contains DOC-01 through DOC-10 with paired Store locations. Requests are deliberately reordered and include DOC-00 and DOC-11. All twelve inputs are nonblank, so the original no-match marker is sufficient for this specific source. A missing-input guard would also be acceptable, but it would not be exercised by these inputs. State which edge cases a dataset actually tests rather than claiming every branch has been demonstrated.

Build the original exact formula and reference uniqueness count. Then choose one interior match, the first reference key, the last reference key, and both absent keys for manual traces. DOC-04 is paired with Store-B2; DOC-01 with Store-A1; DOC-10 with Store-E2. The absent keys must not inherit the first or last reference value. This small case reinforces the same contract without adding new function behavior.

Use the original reverse-order test on complete F:G rows. Because every reference key is unique, all twelve output values remain unchanged. Do not reverse the input list at the same time. If you compare only how many results are registered, a swapped location could go unnoticed. Compare each requested code and output pair or use the stable input row as the comparison key for this fixed test.

Now compare the original and expanded cases conceptually. DOC has two fields in its reference and no missing request. Archive has four fields, a repeated request, two absent keys, and one missing request. Both require exact key membership and paired return values. The expanded diagnostic cases add duplicate reference keys and missing return values, which were not present in the original foundation. A broader lesson does not retroactively change the original source facts.

Write a short transfer note stating what stays the same and what changes between layouts. Stable elements include exact matching, anchored reference ranges, visible no-match handling, uniqueness checks, and complete-row reorder tests. Changed elements include input address, key range, return range, table-relative column index, and whether a missing-input branch is exercised. This distinction is essential when adapting a verified formula to a new worksheet.

```
B2: =XLOOKUP(A2,$F$2:$F$11,$G$2:$G$11,"UNREGISTERED",0)
```

```
H2: =COUNTIF($F$2:$F$11,F2)
```

### E16-X1 Audit the original document-code foundation — practice

Complete and audit the original twelve outputs and ten uniqueness checks. Reverse the complete reference rows in a test copy. Provide a transfer note comparing DOC and Archive without mixing their addresses.

Submit: The original output sequence, uniqueness and reorder results, and a layout-transfer note.

# 17 Independent kit-registry practical

Create a new sheet named Capstone and keep it separate from the guided Archive examples. Import assets/capstone_requests.csv at A1 and assets/capstone_reference.csv at H1. The request data occupies A2:B13; the reference data occupies H2:K9. The field order is the same as Archive: RequestID, RequestedCode, and then Code, Title, Location, Owner. Create output columns C:G and a reference-count column L using the same meanings as before.

This fictional kit registry is a new source, not a renaming exercise. Its keys are sparse KIT identifiers and its locations are Bay values. The source contract requires unique reference codes and nonblank title, location, and owner fields. Requests may repeat. Treat codes as text and preserve their leading zeros. The input files contain source facts only; derive the outputs from the supplied records rather than borrowing the Archive result list.

Choose either the XLOOKUP pathway or the exact VLOOKUP and IFNA pathway, and identify which one you used. Build guarded locations, request match counts, decisions, titles, and owners for every request. If you use VLOOKUP, state the table-relative index for each attribute. Perform a reference uniqueness check and inspect the return fields for completeness before calling a single-match request READY.

Prepare an exception register without editing the raw requests. Distinguish a missing input from a supplied unregistered identifier. Reconcile every request into one of the declared decision states. Explain why a repeated requested code is retained, and trace that code's outputs to the same reference record on both request rows. A correct count reconciliation cannot substitute for a correct row-level mapping.

Run three controlled tests in a copy. First reverse all complete reference records and compare every result while leaving requests fixed. Second change the location for KIT-023 from its supplied value to Bay-D9 and predict all affected RequestIDs before observing the outputs; restore it afterward. Third temporarily append a second KIT-031 reference row with title Cable plan duplicate, location Bay-X9, and owner Technical at H10:K10. Extend all diagnostic and lookup boundaries to row 10, identify the ambiguity, and do not publish a trusted location for that key. Restore the reference and ranges afterward.

The final practical submission should contain the baseline report, the full row-level mapping, the exception register, the uniqueness and completeness audit, and the three test logs. Keep experimental values out of the final baseline. A report with all displayed locations correct but no evidence of exact mode or source boundaries is incomplete because it cannot be

reproduced reliably. Conversely, a visible unresolved exception can be correct when it faithfully represents an absent or missing key.

### E17-X1 Independent kit-registry practical — practice

Complete the entire Capstone report independently. Include all twelve location/title/owner outputs, request counts and decisions, eight reference counts, exception classifications, complete-row reversal comparison, KIT-023 update/restoration, and the extended-range duplicate KIT-031 test with a gated final decision.

Submit: A reproducible baseline workbook and three separate controlled-test logs; do not overwrite source CSVs.

# 18 Build a defensible lookup handover

A lookup report is ready for another learner to review when its method and limitations can be reproduced from the workbook and source files. The handover should name the source snapshot, request and reference ranges, identifier representation, chosen function pathway, exact-match setting, exception meanings, and diagnostic tests. These are not decorative notes; each one defines a condition under which the displayed result is meaningful.

Separate three kinds of evidence. Source evidence shows the original key and paired attributes. Formula evidence shows the request reference, lookup boundary, return boundary or table index, and exact mode. Test evidence shows that selected changes produce the predicted effects and that the baseline was restored. A screenshot of a plausible rack supplies little formula evidence. A formula string without its reference snapshot supplies little source evidence.

Use a fault-to-test map rather than a generic statement that everything was checked. A drifting reference range is detected by inspecting copied endpoints and tracing a first-row key. A locked request cell is detected by comparing several requests. Duplicate keys are detected by reference counts. An offset return array is detected by paired-row traces. A stale append boundary is detected by a request for the newly added key. Missing return fields require a completeness check separate from membership.

No single test proves every property. Full-row reorder invariance is useful, but identical duplicate return values can survive it. Agreement between XLOOKUP and VLOOKUP is useful, but both can use the same short range. A state total can reconcile while two locations are swapped. Strong verification combines independent checks whose failure modes differ. The aim is not to accumulate many similar screenshots; it is to cover distinct ways the mapping can become wrong.

A reviewer should be able to distinguish a source issue from a formula issue. MAP-108 absent from the baseline is a legitimate source-membership exception. MAP-108 appended below a stale endpoint is a boundary mismatch. MAP-107 listed twice with conflicting racks is a reference-uniqueness issue. MAP-302 paired with a shifted return array is a formula-alignment defect. Clear causal labels make corrections smaller and safer than repeatedly rewriting the entire workbook.

Finish with an unresolved-items statement. Do not assert that an unregistered item does not exist anywhere or that a missing location authorizes a default. State which inputs remain unresolved against the declared snapshot and what kind of evidence would resolve them. Retain original input values even when a later approved correction is recorded separately. A trustworthy lookup report can show both successful retrievals and accurately classified uncertainty without pretending that every question has been settled.

### E18-X1 Build a defensible lookup handover — practice

Write a handover for the Capstone baseline. Include exact ranges, formulas or function settings, identifier rules, exception counts and meanings, test coverage, restoration status, and unresolved items. Add a five-entry fault-to-test map using distinct defects from this course.

Submit: A concise but complete method statement and diagnostic map suitable for an independent reviewer.

# Assessment

Choose one best answer per question. Questions 1–10 check the foundation concepts; questions 11–20 apply the extended study units. Complete and submit the practical tasks separately. A strong quiz score does not replace correct practical work.

**1. The original DOC request list has a different order from the reference table. Which field should connect each request to its storage record?**

A. The document code suffix converted to a nearest numeric value

B. The request row number paired with the same reference row number

C. The storage location used as a substitute for the requested identifier

D. The complete document code in the request and reference

**2. For =XLOOKUP(A2,$F$2:$F$11,$G$2:$G$11,"UNREGISTERED",0), what does the final 0 specify?**

A. Search from the last reference row to the first

B. Return zero when the location is empty

C. Choose the nearest registered code if A2 is absent

D. Require an exact match rather than a near match

**3. DOC-00 is absent from the original DOC-01 through DOC-10 reference. What is the correct result under the supplied rule?**

A. Store-A1, because DOC-01 is the first available key

B. Store-E2, because a default location prevents blank output

C. UNREGISTERED, with DOC-00 retained for review

D. MISSING CODE, because no reference row matched

**4. Why should the original code DOC-01 retain its zero when copied into a request?**

A. The zero makes the location calculation more accurate

B. Removing the zero makes exact matching automatically approximate

C. The complete representation belongs to the identifier contract

D. All identifiers must always contain exactly two digits

**5. A reference contains the same document code twice with two different locations. A normal first-to-last exact XLOOKUP returns one location. What is the main risk?**

A. The first-match result can hide an ambiguous reference mapping

B. The formula must always display #N/A for any duplicate

C. The function automatically merges the two location strings

D. The duplicate request is automatically removed from the report

**6. The supplied reference is unsorted. The formula uses exact match mode 0 and ordinary first-to-last searching, not binary search. Which statement is correct?**

A. The location column must be sorted before the lookup can work

B. Sorting only the key column is a safe way to improve reliability

C. Exact matching requires keys to be sorted whenever there are ten rows

D. The supplied unsorted order is valid for this exact-search setup

**7. Which version of the original legacy formula explicitly requests exact matching?**

A. VLOOKUP(A2,$F$2:$G$11,2,TRUE)

B. VLOOKUP(A2,$F$2:$G$11,2)

C. VLOOKUP(A2,$F$2:$G$11,2,FALSE)

D. VLOOKUP(A2,$F$2:$G$11,1,TRUE)

**8. A row shows UNREGISTERED after a correctly configured exact lookup. What conclusion is justified?**

A. The item cannot exist in any archive or future reference version

B. The supplied nonblank key was not found inside the declared reference range

C. The reference definitely has duplicate keys

D. The matching reference record has an empty location

**9. Why does the original practical reverse complete reference rows while leaving requests unchanged?**

A. To make DOC-00 become a valid alias of DOC-01

B. To check that results follow key-to-location mappings rather than row positions

C. To remove duplicate requests by changing their order

D. To prove that no future reference version can contain duplicates

**10. A request looks like a misspelled code, but no correction rule or verified replacement is supplied. What should the learner do?**

A. Select the nearest code to reduce the exception count

B. Overwrite the raw input with the first plausible key

C. Retain the raw value, flag it, and seek correction under a defined rule

D. Delete the request because it cannot be resolved immediately

**11. R04 and R11 both request MAP-107, while the reference contains MAP-107 once. What should the reference-match count show for each request?**

A. 2, because there are two requests

B. AMBIGUOUS KEY, because any repeated text is ambiguous

C. 0, because request repetition invalidates the key

D. 1, because each request finds one reference record

**12. The Duplicates reference contains MAP-107 → Rack-A7 before MAP-107 → Rack-Z9. After reversing complete rows, the raw output changes to Rack-Z9. What is the best final decision?**

A. Trust Rack-Z9 because it is now first

B. Use AMBIGUOUS KEY because the match count remains 2

C. Trust Rack-A7 because it was first in the original file

D. Use UNREGISTERED because the outputs disagree

**13. RET-02 exists once in ReturnQuality but its source location is empty. Which statement correctly distinguishes the issue?**

A. It is a registered key with a missing return field, so use the explicit MISSING LOCATION helper

B. It is unregistered because no visible location was supplied

C. It should borrow the preceding location because its key is valid

D. The key should be deleted to make all registered records complete

**14. TextKeys defines text 0042 → Bin-Zero and text 42 → Bin-Short. What happens conceptually if a learner pads the valid key 42 to 0042?**

A. The learner substitutes another registered identifier and can retrieve the wrong bin

B. The reference automatically combines both records

C. The lookup becomes approximate even with mode 0

D. Only the display changes, so the requested identity is unchanged

**15. Archive uses VLOOKUP(B2,$H$2:$K$11,3,FALSE). Why is the return index 3 correct for Location?**

A. Location is the third worksheet column

B. The index specifies three possible matches

C. There are three location states in the report

D. Location is the third column inside the selected H:K rectangle

**16. A faulty exact lookup pairs H2:H11 with J3:J12. Both arrays have ten cells. Why can it still return the wrong rack?**

A. Exact mode requires a sorted return column

B. A ten-cell array can only be used with ten requests

C. Equal lengths do not preserve the original key-to-attribute row alignment

D. The no-match marker forces the return array down one row

**17. MAP-107 Location changes from Rack-A7 to Rack-A9 in the reference, with its key unchanged. Which baseline requests should change location?**

A. Every request owned by Intake

B. R04 and R11 only

C. R12 only, because MAP-108 is nearby

D. No request, because exact keys cannot have updated attributes

**18. MAP-108 is appended at H12:K12, but formulas still use H2:H11 and J2:J11. R12 remains UNREGISTERED. What should be checked first?**

A. Whether MAP-108 should be replaced by MAP-107

B. Whether lookup, return, and diagnostic ranges all include the new reference row

C. Whether exact mode should be changed to next smaller

D. Whether the request list should be sorted

**19. A legacy lookup uses return index 5 in the four-column H:K table. Why should this not be reported as UNREGISTERED?**

A. The structural #REF! error concerns the formula, not absence of the requested key

B. Every error proves that a key is registered

C. IFNA automatically repairs the table width

D. An out-of-range index asks Excel to use the last valid column

**20. A complete-row reversal leaves all outputs unchanged. Why is a separate reference uniqueness audit still needed?**

A. Duplicate rows with identical returned values can survive the reorder test without changing outputs

B. Unchanged outputs prove that every key occurs exactly once

C. Reversal always removes duplicates automatically

D. The audit is needed only when the reference is sorted

### Foundation completion criteria

Practical pass: all 10 registered codes return their correct locations; both absent codes return UNREGISTERED rather than approximations; uniqueness and reorder checks are documented. Quiz target: 8 of 10.

## Sources and further reading

Microsoft Excel: XLOOKUP function
https://support.microsoft.com/en-us/excel/functions/xlookup-function

Microsoft Excel: VLOOKUP function
https://support.microsoft.com/en-us/excel/functions/vlookup-function

Microsoft Excel: IFNA function
https://support.microsoft.com/en-us/excel/functions/ifna-function

Microsoft Excel: Keeping leading zeros and large numbers
https://support.microsoft.com/en-us/excel/keeping-leading-zeros-and-large-numbers

Microsoft Excel: EXACT function
https://support.microsoft.com/en-us/excel/functions/exact-function

Microsoft Excel: COUNTIF function
https://support.microsoft.com/en-us/excel/get-started/use-the-countif-function-in-microsoft-excel