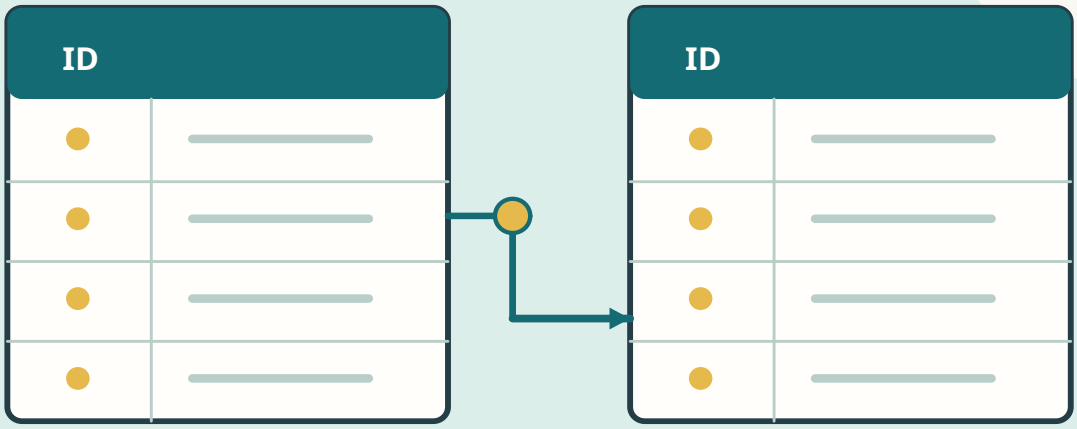
SQL RECORD FILTERING

Filtering Records with SQL Conditions



WHERE / CONDITIONS

AI & DATA

Filtering Records with SQL Conditions

Learning outcomes

- Translate date, category and missing-value rules into three executable filters.
- Distinguish NULL, empty text and whitespace-only text.
- Test inclusion and exclusion at each boundary.
- Translate independent time, category, quantity, and missingness requirements into explicit row predicates.
- Explain AND/OR grouping through distinguishing counterexamples and exact membership evidence.
- Apply half-open normalized UTC timestamp windows and their bounded complements.
- Distinguish SQL NULL, empty text, ASCII spaces, tab, and nonbreaking space without modifying source values.
- Build independent predicted-versus-observed grids and demonstrate that they reject a plausible faulty filter.
- Transfer the method to an allocation task with mixed time and quantity endpoint conventions.

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

Define the population before using operators

A condition is a decision rule applied to a record, not a search phrase. Start by writing the reporting period, accepted categories and missing-value definition independently. The service desk uses a half-open October interval: the first instant on October 1 is included and the first instant on November 1 is excluded. This avoids guessing the last possible time of the month. The supplied timestamps all use the same sortable year-month-day format and UTC convention. That local data contract permits direct text comparisons here; arbitrary date strings from different formats would not justify the same approach.

The category rule is exact. Hardware and Software are two permitted values for the second extract, while Access is not. Do not add status = 'Open' simply because the team is a service desk. Closed requests are still within the specified historical population. An attractive business assumption can be a silent error when it changes the question.

Write a small decision table before writing the query. Include a record just before the start, at the start, just before the end and at the end. Add category and missingness changes one at a time. This makes a failed check informative: you can tell whether the error belongs to the date boundary, category set or missing-value rule rather than treating the whole query as a black box.

Represent missing values deliberately

SQL NULL means an absent or unknown value. Empty text is a stored string of length zero, and a string containing spaces is another stored value. The business contract may group these together, but the database does not make that decision automatically. In this exercise, a missing assignee is either NULL or text whose removal of ordinary surrounding spaces leaves an empty string. Use IS NULL for the former and TRIM(assignee) = '' for the latter.

The practice table deliberately contains a tab character. Default TRIM does not mean remove every conceivable whitespace character. The supplied policy only counts ASCII spaces, so the tab-only record remains nonmissing for this assignment. A production policy might choose a broader normalization rule, but that would be a different requirement needing explicit implementation and tests. Do not silently broaden the definition to make the data look tidier.

A comparison such as assignee = NULL does not test for missingness correctly. Unknown comparisons do not become true merely because their text looks intuitive. Similarly, assignee <> '' does not by itself give a complete nonmissing rule: it needs the NULL and space policy considered. Form the two complementary definitions explicitly, then compare their counts or record sets to ensure each row lands on the intended side.

Combine conditions without widening the request

Use AND when all requirements must hold and OR when either alternative can satisfy a subrule. Parentheses make the intended grouping visible. For example, the missing-value alternatives should sit inside one parenthesized unit before they are combined with the date and category conditions. Without clear grouping, an OR branch may admit records from the wrong month. Operator precedence is not a substitute for a readable statement of business intent.

For a small category list, IN ('Hardware','Software') expresses membership directly and avoids repeating the field name. For the outside-October query, the date condition is received_at < the start OR received_at >= the end. Keep that pair grouped before adding Hardware and missingness. This is a useful example of complement logic: a half-open interval has two outside regions, not merely dates earlier than the start.

Build each query incrementally. First inspect dates and IDs, then add the category condition, then the missingness condition. Save a temporary record count after each step during your own reasoning. The final submission needs the completed extracts, but these intermediate checks help locate mistakes. Never remove difficult cases from the input table to make the output match expectations. A boundary row is evidence for the test, not an obstacle to delete.

Use boundary evidence rather than intuition

The practical assessment uses eight named records as boundary cases for every extract. Predict inclusion or exclusion before executing the final SQL, and record the reason in ordinary language. Then compare the observed membership of each ID against your prediction. A correct overall count is insufficient: one wrongly included row and one wrongly excluded row can cancel each other numerically.

Make the test grid independently from the query. If you copy the query's own logic into a second query and call that a test, both can share the same error. Reading the raw timestamp, category and assignee representation for eight records is manageable and provides a useful independent check. For invisible text, consult the dataset notes or a diagnostic expression such as LENGTH(assignee); a blank-looking cell is not reliable evidence of NULL.

Deliver three SQL files or three clearly named statements, each with stable output ordering. Include the decision grid and a short missingness note that mentions the tab rule. Explain any discrepancy you corrected, but do not claim that the supplied artificial data proves the query works for all date formats or all international text. The result demonstrates a bounded, tested interpretation of one explicit contract. That precision is more valuable than a vague claim that the filter handles every case.

Practice protocol and submission standard

All organizations, identifiers, transactions and observations in this pack are fictional. Use only the supplied local database, not a workplace connection. Open it through the read-only runner in the practice folder. Save your SQL in a separate UTF-8 text file and retain the result headers. A query that returns plausible numbers is not automatically correct: compare the output against the stated population, grain, field order and edge cases. Keep the original database unchanged. The exercise asks for evidence of a repeatable method, not a screenshot alone. Submit the SQL, the exported result and a short explanation of checks. This practice course is not an accredited qualification.

Worked example

Title: A half-open interval rehearsal

Sql

```
WITH demo(id,at) AS (VALUES ('A','2026-04-01 00:00'),('B','2026-05-01 00:00'))
    SELECT id FROM demo WHERE at >= '2026-04-01 00:00' AND at <
    '2026-05-01 00:00';
```

Explanation

Only A is returned. The start belongs to the interval and the following month does not. This rehearsal uses a different month from the assessed extracts.

Foundation practice

Scenario

A fictional service desk needs three request extracts. Timestamps are normalized UTC text in YYYY-MM-DD HH:MM format; assignee is NULL, empty, whitespace-only or a nonblank team code.

Database file: 3H-AI-0142_practice/practice.sqlite

Schema file: 3H-AI-0142_practice/schema.sql

Runner file: 3H-AI-0142_practice/run_query.py

Readme file: 3H-AI-0142_practice/README.txt

Tables

requests

request id	received at	category	assignee	status
R01	2026-09-30 23:59	Hardware	NULL	Open
R02	2026-10-01 00:00	Hardware	NULL	Open
R03	2026-10-31 23:59	Hardware		Open
R04	2026-11-01 00:00	Hardware	NULL	Open
R05	2026-10-15 09:00	Hardware		Open
R06	2026-10-15 09:00	Hardware	Team-A	Open
R07	2026-10-15 09:00	Software	NULL	Open
R08	2026-10-15 09:00	Software	Team-B	Open
R09	2026-10-20 10:00	Access		Open
R10	2026-10-20 10:00	Access	Team-C	Open
R11	2026-09-15 12:00	Software	Team-B	Closed
R12	2026-11-15 12:00	Hardware	Team-A	Open
R13	2026-10-01 00:01	Software		Closed
R14	2026-10-31 00:00	Hardware	Team-D	Closed
R15	2026-10-22 11:00	Hardware		Open
R16	2026-09-30 23:59	Hardware		Closed

Csv file: requests.csv

Requirements

Write three queries, each returning request_id, received_at, category, assignee ordered by request_id. Q1: October Hardware requests with missing assignee. Q2: October Hardware or Software requests with a nonmissing assignee. Q3: Hardware requests outside October with missing assignee. October means >= 2026-10-01 00:00 and < 2026-11-01 00:00. Missing means NULL, empty string or only ASCII spaces; a tab is explicitly not missing under this contract. Status is not a filter. Test the eight cases R01-R08 for every query and retain a predicted-versus-observed grid.

Null note

JSON null and SQL NULL mean missing. CSV uses <NULL> as a display marker; empty text remains empty text. No import is required.

Run instructions

Open the practice folder, save one SELECT or WITH ... SELECT statement as my_query.sql, then run python3 run_query.py my_query.sql (or py run_query.py my_query.sql on Windows). The runner opens the supplied SQLite database read-only.

Submission

Saved SQL statements, exported query outputs, specified validation evidence and a short interpretation. Keep the database unchanged.

Blank worksheet: 3H-AI-0142_practice/boundary_test_blank.csv

Foundation tasks

- Exercise 1: Inspect all assignee representations and draft the missing/nonmissing rule without changing any data.
- Exercise 2: Predict membership for R01-R08 in all three extracts and explain each date boundary.
- Capstone: Submit all three contract-compliant queries, exported results and the 24-cell predicted-versus-observed membership grid.

01 Specify a row decision before writing a condition

The fictional Bracken Nursery receives consignments of seeds, tools, and pots. Each row of nursery_receipts in assets/workshop.sqlite represents one receipt record, not one pack. The table contains twenty-two records with arrival timestamps, item classes, optional bay codes, pack quantities, and inspection stages. The task is to select the right records without changing any source data. All names and events are invented for local practice.

The nursery needs three extracts for a defined receiving week. G1 identifies Seeds receipts during the week with a missing bay code. G2 identifies Seeds or Tools receipts during the week with a nonmissing bay code. G3 identifies Seeds receipts outside the week with a missing bay code. Every extract returns receipt_id, arrived_at, item_class, bay_code in that order, sorted by receipt_id. Neither packs nor inspection_status is an eligibility rule for these three extracts.

Contract component	Local definition
Receiving week	2026-11-09 00:00:00 inclusive to 2026-11-16 00:00:00 exclusive, UTC.
Timestamp representation	All values are non-NULL normalized text: YYYY-MM-DD HH:MM:SS.

Contract component	Local definition
Missing bay code	NULL, empty text, or text containing only ordinary ASCII spaces.
Not missing under this policy	Tab-only or nonbreaking-space-only text.
Source boundary	The supplied fictional local database only.

A filter is a decision applied separately to each source row. Write its dimensions before choosing operators: time window, permitted category, and missingness rule. This prevents a loose phrase such as incomplete deliveries from accumulating unstated assumptions. A Checked receipt can still lack a bay code, and a zero-pack record can still belong to the receiving history. The named contract, rather than an intuitive picture of nursery work, determines inclusion.

The source assets include complete CSV companions, a data dictionary, and representation_notes.json for invisible text. No import is required. Use the database and read-only runner as supplied. The smaller gate_demo table provides worked examples with the same field meanings but different records. Learn on that table, then write the main extracts independently. Do not copy a result ID list into a WHERE clause as a substitute for implementing the business rule.

Before executing a filter, predict a few decisions from raw facts. N02 is exactly at the beginning of the week; N04 is exactly at the next week boundary. These two records differ in time membership even though their item class and missing bay state match. Deliberately contrasting near-identical cases makes a boundary defect easier to find than inspecting a random sample of ordinary middle-of-week records.

E01-X1 Write the three decision contracts

Write G1, G2, and G3 as plain-language rules listing each required condition. Identify two source fields that must not become filters. State what one output row represents and what the timestamp comparison assumes.

Submit: Three bounded decision rules and a source-assumption note.

02 Compare a single field and test both sides of the boundary

Start with one condition whose meaning is easy to check. Equality asks whether a field matches a specified value; a numeric comparison asks whether a measured count falls on one side of a boundary. The difference between > and >= matters at the exact threshold. A statement that asks for at least eight packs includes eight, while more than eight does not. Translate the language into an operator before inspecting the output.

The gate_demo table contains pack quantities 4, 8, 6, 12, 0, and 7. The next worked query selects records with at least eight packs. It returns D2 at the threshold and D4 above it. D6 has seven, so it is excluded. This small comparison makes the operator visible without mixing

it with category or date rules. Record IDs remain in the result so each decision can be traced back to its source facts.

```
SELECT receipt_id, packs FROM gate_demo WHERE packs >= 8 ORDER BY receipt_id;
```

Verified worked result

receipt id	packs
D2	8
D4	12

A numeric zero is an observed value, not a missing value in these fixtures. It fails a positive threshold because of its quantity, not because SQL treats it as unknown. Conversely, a nullable field such as bay_code needs an explicit missingness rule rather than a numerical threshold. Choose a predicate that matches the field meaning. Do not compare text identifiers with numeric values merely because some identifiers contain digits.

Write exact category text in single quotes. The predicate item_class = 'Seeds' compares the field with that literal category. A bare word Seeds would be interpreted as a name rather than the intended text value. The supplied category vocabulary is exact and controlled; there is no requirement for approximate matching, case folding, or spelling correction. Those would be separate transformations requiring their own rules and tests.

For every threshold, select three useful test facts: one below it, one exactly equal, and one above it. A query can look correct on distant values yet fail at equality. For a closed numeric range, both endpoints belong. For the receiving-week timestamp interval, only the lower endpoint belongs. Do not assume all ranges use the same boundary convention. State the convention next to the business requirement.

E02-X1 Implement an inclusive quantity band

On nursery_receipts, select receipt_id and packs for quantities from 8 through 12 inclusive, ordered by receipt_id. This is a separate quantity drill, not a change to G1–G3. Identify one included lower-bound record, one included upper-bound record, and an excluded value just above the band.

Submit: Saved query, exact output, and three boundary observations.

03 Use AND for requirements that must all hold

AND means a record must satisfy every connected requirement. If the nursery asks for Seeds receipts with at least eight packs, a Tools receipt with eight packs does not qualify, and a Seeds receipt with four packs does not qualify. The condition is not a score in which one strong match compensates for another failure. It is an intersection of required properties.

The following gate_demo query combines two familiar predicates. D1 is Seeds but has only four packs. D2 has eight packs but is Tools. D4 has twelve but is Pots. None of those partial matches is sufficient. This worked query intentionally returns no rows, while still reporting its

two column headings. An empty result can be the correct consequence of the rules rather than a database failure.

```
SELECT receipt_id, packs FROM gate_demo WHERE item_class='Seeds' AND packs >= 8
    ORDER BY receipt_id;
```

Verified worked result

receipt id	packs

When a result is empty, inspect the decision rather than immediately weakening it. Check that the table exists, field names are correct, and the literals match the supplied vocabulary. Then test each condition separately to understand whether their intersection is empty. Do not replace AND with OR merely to produce a more satisfying-looking list. That would broaden the population without changing the written request.

Seeds?	At least 8 packs?	Both required
Yes	Yes	Include
Yes	No	Exclude
No	Yes	Exclude
No	No	Exclude

A truth table is a concise way to communicate combination logic. It states decisions independently of the order in which the database happens to evaluate expressions. SQL engines may optimize how conditions are processed. Your reasoning should depend on the logical requirements, not on a claim that the first written predicate always runs before the second. In this course all predicates are ordinary read-only expressions with no external side effects.

For a longer filter, group the requirements by meaning: time, category, missingness, and any explicitly requested quantity band. Each group should have a plain-language description. This makes a correction local. If a record at the end timestamp is incorrectly admitted, investigate the time group rather than rewriting the category list or changing data. Clear structure reduces the chance that fixing one defect introduces another.

E03-X1 Find an intersection without broadening it

Select receipt_id, item_class, packs from nursery_receipts where the class is Seeds and packs are at least 12. Do not apply the receiving-week rule to this separate drill. Explain why a Pots record with twenty packs does not qualify.

Submit: Query, complete result, and one exclusion explanation.

04 Group alternatives so they do not escape shared restrictions

OR connects alternatives. A receipt may be Seeds or Tools, but it does not need to be both. The difficulty appears when alternatives share another restriction. If the requirement is at least eight packs and either Seeds or Tools, the quantity threshold must apply to both categories. Parentheses make that grouping visible: the alternative categories form one requirement, which is then connected to the quantity rule with AND.

SQLite gives AND higher precedence than OR. Without parentheses, a clause written as item_class = 'Seeds' OR item_class = 'Tools' AND packs >= 8 means all Seeds records, plus Tools records meeting the threshold. It does not mean both categories are constrained by eight packs. The distinction is substantive: in gate_demo, low-quantity Seeds records would slip through the first branch.

Candidate interpretation	D1: Seeds, 4 packs	D2: Tools, 8 packs	D5: Seeds, 0 packs
Threshold applies to both categories	Exclude	Include	Exclude
All Seeds OR qualifying Tools	Include	Include	Include

The correct worked query below makes the intended alternatives a single parenthesized group. Its result contains only D2. The demonstration is not a request to change the main receiving-week extracts; it isolates the same logical risk that occurs when date conditions are accidentally applied to only one missingness branch.

```
SELECT receipt_id, item_class, packs FROM gate_demo WHERE (item_class='Seeds' OR
    item_class='Tools') AND packs >= 8 ORDER BY receipt_id;
```

Verified worked result

receipt id	item class	packs
D2	Tools	8

Parentheses are also valuable when precedence already gives the intended result. They communicate where a business subrule begins and ends, which makes later editing safer. A reviewer should be able to identify the category alternatives without mentally reconstructing the entire expression hierarchy. Avoid using extra parentheses as decoration around individual words while leaving the important OR combination ungrouped.

To test grouping, choose a record that satisfies the alternative likely to escape but fails a shared restriction. D1 is useful because it is Seeds yet below the threshold. A record satisfying every condition would pass both the correct and incorrect query and therefore would not expose the bug. Strong tests distinguish competing interpretations rather than merely confirming obvious matches.

E04-X1 Design a grouping counterexample

Run a main-table drill selecting Seeds or Tools with packs >= 10. Use a parenthesized category OR or equivalent IN. Identify one Seeds receipt that an ungrouped Seeds OR Tools AND threshold query would wrongly include.

Submit: Correct query, complete result, and one specific counterexample with values.

05 Express category membership and its limited complement

IN provides a compact way to test whether one field belongs to an explicit list of values. item_class IN ('Seeds','Tools') expresses the same category choice as two equality tests connected by OR. It avoids repeating the field name and makes the accepted vocabulary easy to review. A comma inside a single string, such as 'Seeds,Tools', is one value, not a two-item list.

For the supplied non-NULL category field, NOT IN ('Seeds','Tools') selects the remaining category, Pots. That complement is safe to describe here because the dictionary guarantees non-NULL item_class values and the list itself contains no NULL. Do not generalize that result to arbitrary nullable fields or lists containing NULL. Unknown comparisons can prevent a row from satisfying a negated condition.

The following gate_demo query displays the accepted categories without adding quantity, date, or missingness rules. D6 is outside the receiving week but still belongs to this category-only rehearsal. Its presence is not an error. When inspecting intermediate outputs, always identify which part of the contract has been applied so far. Otherwise a useful diagnostic extract can be mistaken for a failed final report.

```
SELECT receipt_id, item_class FROM gate_demo WHERE item_class IN ('Seeds','Tools')
    ORDER BY receipt_id;
```

Verified worked result

receipt id	item class
D1	Seeds
D2	Tools
D3	Seeds
D5	Seeds
D6	Tools

Exact category matching also avoids accidental substring logic. A search for letters appearing somewhere in a description would answer a different question from membership in a controlled category vocabulary. This course does not need LIKE, regular expressions, or

fuzzy matching. Using a more complicated feature is not an improvement when it weakens the connection between the query and the brief.

A useful category check includes one accepted value from each branch and one rejected value. For the main fixture, compare a Seeds receipt, a Tools receipt, and a Pots receipt with otherwise similar facts. If all observed records belong to just one accepted category, a misspelled second category could go unnoticed. Coverage should reflect the alternatives in the requirement, not merely the number of rows inspected.

Do not add a new category because its name sounds related. If the nursery later introduces Soil, the current Seeds-or-Tools extract still excludes it until the recipient revises the rule. A controlled selection list is an explicit business boundary. Treating every unfamiliar category as accepted would make the report silently broader as the source vocabulary grows.

E05-X1 Check a category partition

Produce separate receipt_id/item_class outputs for Seeds or Tools and for Pots, with no other filters. Use the source CSV to verify that their ID lists do not overlap and together cover all twenty-two receipts.

Submit: Two queries, exact outputs, and a complete category-partition statement.

06 Use a half-open timestamp interval with stated assumptions

The receiving week includes its first instant and excludes the first instant of the following week. Write arrived_at >= '2026-11-09 00:00:00' AND arrived_at < '2026-11-16 00:00:00'. This half-open design avoids inventing a final representable moment such as 23:59:59.999999. It also assigns a shared boundary cleanly: a receipt at November 16 midnight belongs to the next week, not both weeks.

Direct text comparison works in these fixtures because every timestamp uses the same year-month-day, hour-minute-second format, zero padding, and UTC convention. The field is non-NULL. The chronological and text ordering therefore align for the supplied values. That is a data contract, not a general claim that SQL recognizes any date-looking string. Mixed timezones, locale formats, and unpadded values require a different preparation problem that this course does not solve.

Timestamp	Inside receiving week?	Reason
2026-11-08 23:59:59	No	Before the inclusive start.
2026-11-09 00:00:00	Yes	Exactly the lower boundary.
2026-11-15 23:59:59	Yes	Still below the next-week boundary.
2026-11-16 00:00:00	No	Exactly the excluded upper boundary.

BETWEEN includes both endpoints. It is convenient for the earlier inclusive numeric band, but using it with the next-week timestamp would also include November 16 midnight. Conversely, a date-only end string such as 2026-11-15 can exclude timestamps later on that day under a text comparison. Use the boundary that represents the intended interval instead of trying to infer the last usable text value.

The gate_demo rehearsal returns the records inside the week before any category or bay rule is applied. Notice that Pots can appear in this diagnostic result. The time predicate should answer only the time question. Later requirements narrow the list. Keeping intermediate stages conceptually separate helps you distinguish a correct partial filter from the final combined extract.

```
SELECT receipt_id, arrived_at FROM gate_demo WHERE arrived_at >=
    '2026-11-09 00:00:00' AND arrived_at < '2026-11-16 00:00:00' ORDER BY
    receipt_id;
```

Verified worked result

receipt id	arrived at
D1	2026-11-09 00:00:00
D2	2026-11-12 10:00:00
D4	2026-11-12 11:00:00
D5	2026-11-12 12:00:00

E06-X1 Predict four exact time boundaries

Select receipt_id and arrived_at for the nursery receiving week, with no other filters. Before running, predict N01, N02, N03, and N04. Compare the observed membership and explain why a next-week BETWEEN endpoint would be wrong.

Submit: Query, full result, and four predicted-versus-observed decisions.

07 Describe the outside region without losing either side

The outside of [start,end) has two parts: times before start and times at or after end. Connect those parts with OR. A query that uses only arrived_at < start finds early records but misses late ones. A query that uses arrived_at > end misses the exact upper boundary, which is outside the original interval. Negation should be derived from the endpoint policy, not improvised from the phrase not this week.

For this non-NULL timestamp field, NOT (arrived_at >= start AND arrived_at < end) gives the same membership as the two explicit outside comparisons. The expanded form often makes boundary review clearer. If timestamps were nullable, a missing timestamp would not automatically become an outside date: the comparison would be unknown. The supplied

non-NULL guarantee matters when you claim the inside and outside populations form a complete partition.

The gate_demo query below returns D3 at the upper boundary and D6 before the lower boundary. Both sides are necessary to describe the outside population. It does not filter category or bay code yet. When constructing G3 later, place the entire outside OR expression in parentheses before adding Seeds and missing-bay requirements.

```
SELECT receipt_id, arrived_at FROM gate_demo WHERE arrived_at <
    '2026-11-09 00:00:00' OR arrived_at >= '2026-11-16 00:00:00' ORDER BY
    receipt_id;
```

Verified worked result

receipt id	arrived at
D3	2026-11-16 00:00:00
D6	2026-11-08 10:00:00

The partition check is useful independent evidence. For the main table, every one of the twenty-two non-NULL timestamps should land in exactly one of the time-only inside and outside lists. The lists should have no shared IDs. This test checks time coverage, not the final G1/G2/G3 relationship, because those extracts have additional category and missingness rules. Do not expect three differently defined business reports to partition the entire source.

A complement claim must name its universe. Outside the week among all receipts differs from outside the week among Seeds receipts with missing bays. In the latter case, category and missingness remain required on both date branches. Write the phrase among Seeds receipts with missing bays next to the time decision to prevent an OR branch from escaping those shared restrictions.

When a filter is revised, preserve the earlier predicted boundary decisions. They are not stale simply because the query text changed. Re-executing the same exact-start and exact-end cases helps confirm that an unrelated repair did not reverse the time policy. Good small tests remain useful across edits because they express the contract rather than a particular SQL spelling.

E07-X1 Verify a time partition

Produce the outside-week receipt_id/arrived_at list. Compare it with E06 inside-week output: state both counts, verify no overlap, and identify every outside ID.

Submit: Outside query and a complete two-list reconciliation.

08 Understand why NULL does not behave like an ordinary value

SQL NULL represents absence or unknown information, not the literal word NULL. Comparing a value with NULL using = does not produce an ordinary true equality test. Use IS NULL to identify the missing marker and IS NOT NULL to identify a present value. An empty string is present text of length zero, so IS NOT NULL alone does not mean a bay code is operationally usable under the nursery's local policy.

A WHERE condition keeps rows when the condition evaluates true. False and unknown do not qualify. This distinction explains why bay_code <> '' is not a complete policy for assigned bays. It does not include NULL values, but it does include strings containing spaces, a tab, or a nonbreaking space. The desired business meaning needs the explicit NULL and ASCII-space handling that later units combine.

Source bay code	bay code IS NULL	bay code = empty text	Interpretation
NULL	True	Unknown	No stored value.
Empty text	False	True	Stored zero-length string.
Three spaces	False	False	Stored text, even if it looks blank.
Bay-A	False	False	Ordinary populated code.

The worked query uses IS NULL on the small teaching table. It returns D1. D3 has empty text and D4 has spaces, so neither is SQL NULL. The distinction is not a defect in the database. It tells you that several stored representations may need to be grouped by the reporting policy, and that grouping should occur in a query rather than by rewriting the source.

```
SELECT receipt_id, bay_code FROM gate_demo WHERE bay_code IS NULL ORDER BY
    receipt_id;
```

Verified worked result

receipt id	bay code
D1	NULL

Do not rely on a blank-looking cell to diagnose which representation is present. A viewer may display NULL, empty text, and whitespace similarly. Use the source JSON, representation notes, or diagnostic functions to inspect the data. The point is not to memorize every invisible Unicode character; it is to stop assuming that visual emptiness is a reliable database predicate.

A missing-data policy can be broader than SQL NULL without making NULL and text identical. The nursery chooses to treat NULL, empty text, and ASCII-space-only strings as missing bays. It explicitly does not include tab-only or nonbreaking-space-only strings. Whether that policy

would suit another organization is outside this exercise. Your responsibility is to implement and explain the stated rule, including its limitations.

E08-X1 Separate absent values from empty strings

Produce one receipt_id/bay_code list for SQL NULL and another for exactly empty text on nursery_receipts. Use no other filters. Explain why the two lists differ and neither alone implements the full missing-bay policy.

Submit: Two queries, complete results, and an explanation of the omitted representation.

09 Make invisible text visible and keep the trim policy narrow

TRIM(bay_code) removes ordinary spaces from both ends under the default SQLite behavior used here. If the result is empty text, the original string contained no characters other than those spaces, or was already empty. The function does not automatically remove all Unicode whitespace. A tab and a nonbreaking space remain characters after this default trim. That is why the asset notes name the exact code points rather than merely saying whitespace.

The following diagnostic on selected source IDs displays LENGTH and HEX beside the original value. LENGTH counts text characters for these strings; HEX shows their UTF-8 bytes. A NULL input remains NULL for LENGTH, while HEX(NULL) is empty text in the tested SQLite engine. Do not use a blank HEX display alone to classify missingness. Combine the diagnostics with IS NULL and the stated source facts.

```
SELECT receipt_id, LENGTH(bay_code) AS chars, HEX(bay_code) AS utf8_hex,
    LENGTH(TRIM(bay_code)) AS trimmed_chars FROM nursery_receipts WHERE receipt_id
    IN ('N02','N03','N05','N15','N17','N18') ORDER BY receipt_id;
```

Verified worked result

receipt id	chars	utf8 hex	trimmed chars
N02	NULL		NULL
N03	0		0
N05	3	202020	0
N15	1	09	1
N17	1	C2A0	1
N18	7	204261792D4520	5

For these supplied examples, a tab has hexadecimal byte 09, three ASCII spaces have 202020, and the nonbreaking space has C2A0. The latter is one text character encoded with two UTF-8 bytes. This difference is enough to explain why a grid that looks blank can contain different values. You do not need to redesign text encoding to implement the local filter.

N18 contains an ordinary space before and after Bay-E. Trimming its ends produces Bay-E, which is nonempty, so it is nonmissing. The query may use TRIM for the decision while returning the original bay_code value, including its spaces. A filter decides which rows to keep; it need not clean the displayed source field. Preserve that separation unless the recipient explicitly requests normalized output values.

The independent classifier used in checking must follow the same stated policy without copying SQLite's query expression. For a human review, inspect the exact representation notes. In an independent program, removing only the ASCII space character is different from a general whitespace-strip operation. A general strip would classify the tab and nonbreaking space differently and create a false disagreement with the correct query.

An apparent inconsistency can therefore reveal a policy mismatch in the test rather than an SQL defect. Before changing the filter, compare the definitions used by both methods. A test is not automatically authoritative because it was written in another language. It must encode the same business rule while deriving its decisions independently from the raw facts.

E09-X1 Classify the non-obvious bay codes

Classify N05, N15, N17, N18, and N20 under the exact missing policy. Use representation_notes.json and a diagnostic query that returns receipt_id, LENGTH(bay_code), HEX(bay_code), and LENGTH(TRIM(bay_code)). Explain why returning the original value does not undo the filter.

Submit: Five classifications with diagnostic evidence and a filter-versus-transformation note.

10 Construct missing and nonmissing predicates as explicit partners

The nursery's missing predicate has two alternatives: bay_code IS NULL OR TRIM(bay_code) = ''. Put them inside parentheses when combining them with other requirements. The first branch handles the absent value. The second handles empty text and text containing only ordinary spaces. This is a deliberate business grouping of distinct storage representations, not a claim that the source values are identical.

The corresponding nonmissing predicate is bay_code IS NOT NULL AND TRIM(bay_code) <> ''. Both requirements must hold. A code must exist, and removing surrounding ASCII spaces must leave something. The tab-only and nonbreaking-space-only cases qualify under this narrow rule. A reader may find that policy surprising, so mention it explicitly in the handoff instead of implying the output contains only visibly meaningful bay names.

The small worked missing-bay query returns D1, D3, and D4. D5 contains a tab and is therefore not selected. The demonstration has no date rule: D3 remains despite being at the next-week boundary. That is correct for a missingness-only diagnostic. A final business extract must combine the missingness group with its own time and category requirements.

```
SELECT receipt_id, bay_code FROM gate_demo WHERE bay_code IS NULL OR
    TRIM(bay_code)='' ORDER BY receipt_id;
```

Verified worked result

receipt id	bay code
D1	NULL
D3	
D4	

A useful full-source check is that the missing and nonmissing ID sets have no overlap and together cover all twenty-two nursery records. Because the definitions are explicit partners, every supplied representation should fall on one side. Do not validate by count alone: record the IDs, especially tab and nonbreaking-space cases. One mistaken inclusion and one mistaken exclusion can preserve a total while changing the meaning.

This partition does not mean every nonmissing value is a valid physical bay. The contract classifies representation, not existence in a separate reference list. Verifying whether Bay-G exists, whether a code is approved, or whether a tab should be repaired would be additional tasks. Keep your conclusion precise: nonmissing under the stated ASCII-space policy, not confirmed location quality.

When combining the predicate with dates, wrap the whole missing alternative before appending it with AND. Otherwise a row with empty trimmed text may qualify even outside the requested period. The same grouping principle from the earlier quantity example applies here, but the escaped branch is less visually obvious because it concerns blank-looking values.

E10-X1 Check the full missingness partition

Write separate missing and nonmissing receipt_id/bay_code extracts for the entire nursery table. Record complete ID lists, counts, overlap, and coverage. Explicitly locate N15 and N17.

Submit: Two queries and a complete twenty-two-record partition check.

11 Build the main extracts and retain a stage-by-stage explanation

You now have the components needed for G1, G2, and G3. Assemble each filter from its plain-language contract rather than copying the previous query and making unreviewed edits. G1 uses inside-week, Seeds, and missing. G2 uses inside-week, Seeds-or-Tools, and nonmissing. G3 uses outside-week, Seeds, and missing. Each output preserves the original bay_code and returns the four exact requested columns.

A stage trace helps explain why a final population is smaller than the source. Start with all receipt IDs, then apply the time rule, then category, then missingness. These are conceptual checkpoints for review, not a guarantee of the engine's physical evaluation order. You can save intermediate diagnostic outputs if they help locate an error, but label them so a reviewer does not confuse them with the final deliverables.

Extract	Time group	Category group	Bay group
G1	Inside receiving week	Seeds only	Missing
G2	Inside receiving week	Seeds or Tools	Nonmissing
G3	Outside receiving week	Seeds only	Missing

The three extracts are intentionally not an exhaustive partition of all receipts. Pots records do not belong to any of them. An outside-week Seeds receipt with a nonmissing bay also belongs to none. G1 and G2 differ in category as well as missingness, so their union is not simply all inside-week Seeds-or-Tools records. Avoid forcing totals to reconcile to a universe the reports were never designed to cover.

Output ordering is a review aid. Use ORDER BY receipt_id so the same rows appear in a stable sequence when another learner reruns the query. Sorting does not determine eligibility. Keep the ID in the result so an inclusion can be inspected against its raw timestamp, category, and bay representation. A report containing only category names would obscure the specific records you need to validate.

Do not filter inspection_status unless the contract changes. N03 and N05 are Checked but can still satisfy a missing-bay report. N14 is Checked and may belong to the nonmissing report. A status field often invites an intuitive extra condition because it seems related to work completion. Here it is included in the fixture specifically to test whether you respect the written population rather than add a convenient assumption.

E11-X1 Write all three receiving-week extracts

Create G1, G2, and G3 against nursery_receipts, with receipt_id, arrived_at, item_class, bay_code and stable ID order. For G1 record the number of IDs after the time stage, after adding Seeds, and after adding missingness. Keep inspection_status out of the filter.

Submit: Three SQL files, complete results, and a three-stage G1 trace.

12 Test a deliberately broken predicate with counterexamples

A strong test suite should reject a plausible wrong query, not merely agree with the query you hope is correct. Consider a candidate for G1 that appends OR TRIM(bay_code)='' outside the intended missingness parentheses. Because AND binds more tightly than OR, the empty-after-trim branch can admit records regardless of their date or item class. This candidate can return convincing Seeds matches while also leaking unrelated rows.

Use source facts to predict counterexamples before executing a faulty candidate. N09 is Pots, inside the week, and empty bay text. N19 is Pots, outside the week, and empty bay text. Both fail the G1 contract, but both can slip through the ungrouped trim branch. N16 is Seeds with empty text before the week, so it tests the date restriction separately from the category restriction.

Counterexample	Contract reason for exclusion	What an escaped OR branch ignores
N09	Pots is not Seeds.	Category
N16	Before receiving-week start.	Time
N19	Outside week and wrong category.	Both time and category

Keep a broken query in a separately labeled diagnostic file, never as a final answer. The goal is to understand the failure and show that your independent decisions detect it. Do not alter source rows to make the candidate pass. A failed assertion is useful evidence when it identifies a real divergence between the business contract and the implementation.

Counts alone can reveal some broken candidates but are not a complete defense. A different defect could admit one wrong row and exclude one correct row, leaving the total unchanged. Compare ID membership and then values. If the correct expected set is derived manually from the raw source facts, a shared SQL precedence mistake is less likely to infect both the implementation and its test.

After fixing parentheses, rerun the boundary and representation cases, not just the newly discovered counterexample. The repaired predicate should still include the exact start, exclude the exact end, retain Checked status where appropriate, and classify tabs according to policy. A narrow repair should restore the intended rule without changing unrelated dimensions of the selection.

E12-X1 Show that your tests catch a grouping defect

Evaluate this faulty G1 condition in a separate read-only candidate: inside-week AND Seeds AND bay_code IS NULL OR TRIM(bay_code)=''. Compare its IDs with your correct G1. List every extra ID and explain two distinct violated restrictions.

Submit: Faulty-candidate query/output, exact extra-ID list, and corrected-query evidence.

13 Create a predicted-versus-observed membership grid

The asset boundary_predictions.csv lists all twenty-two receipt IDs with separate prediction, observation, and reason fields for G1, G2, and G3. Fill predictions from the source facts before copying observed membership from query outputs. This produces sixty-six inclusion decisions. The purpose is not paperwork volume: each decision ties a rule to a named record and makes discrepancies locatable.

For a concise written review, emphasize cases that isolate one dimension. N01 through N04 test the four time boundary positions. N05 tests ASCII spaces; N06 a normal code; N07 and N08 the second accepted category; N09 and N10 a rejected category. N15 and N17 test invisible nonmissing text. N21 tests whether an unrequested positive-quantity condition has been added. The full grid then checks that the remaining records were not accidentally overlooked.

A reason should name the failed or satisfied requirement. Write outside at upper boundary rather than does not match. A row can fail more than one requirement; mentioning the decisive reason is enough for a simple explanation, but note multiple failures when diagnosing an escaped OR branch. Keep prediction independent of observation: if you revise a mistaken prediction after studying the policy, preserve a short explanation rather than silently replacing it.

For each extract, compare the complete expected and observed ID lists in the same order. If they differ, identify missing IDs and extra IDs separately. This distinction helps locate a narrow-versus-broad filter error. Then compare the returned source values and exact headers. Membership correctness does not excuse returning a normalized bay value when the contract asks to preserve the original field.

The grid supports a bounded conclusion: these statements match these supplied facts under this explicit policy. It does not prove correctness for mixed timestamp formats, unspecified timezones, new categories, or a broader whitespace definition. State those assumptions in the submission note. Precision about the tested universe makes the evidence stronger, not weaker.

Do not share the instructor key with a learner before they make their predictions. A copied expected-ID list can make a grid look complete without demonstrating any understanding. A reviewer should be able to ask why N15 belongs to G2 or why N04 belongs to G3 and receive an explanation grounded in the source representation and endpoint rules.

E13-X1 Complete and explain the full decision grid

Complete all sixty-six G1/G2/G3 decisions in boundary_predictions.csv. For N02,N04,N05,N09,N15,N17,N21, include specific source-based explanations. Compare to actual query membership and resolve every discrepancy without editing the database.

Submit: Completed prediction/observation grid and a discrepancy note, even if no discrepancy remains.

14 Transfer filtering to a bounded materials-allocation request

The separate allocations table contains twelve fictional requests for compost, mulch, and gravel. One row is one request, not one tray. The coordinator needs requests during the first December receiving week, for Compost or Mulch, requesting four through eight trays inclusive, and lacking a pickup code under the same NULL/empty/ASCII-space policy. Return allocation_id, requested_at, material, tray_count, pickup_code, ordered by allocation_id. State is not a filter.

The time interval is 2026-12-01 00:00:00 inclusive to 2026-12-08 00:00:00 exclusive. Notice the combination of boundary conventions: time uses an exclusive upper endpoint, while the tray band includes both four and eight. Do not reuse one operator pattern for both ranges merely because both appear in the same sentence. A correct filter must preserve each dimension's stated meaning.

Record excerpt	Time or category fact	Quantity or code fact
A01	Exactly December 1 start; Compost	4 trays; NULL pickup code.
A04	Inside week; Compost	9 trays; NULL pickup code.
A06	Exactly December 8 start; Compost	5 trays; NULL pickup code.
A10	Inside week; Mulch	8 trays; tab-only pickup code.
A11	Inside week; Mulch	8 trays; NULL pickup code.

The asset data_dictionary.json and allocations.csv contain every input needed for the independent task. Use them to predict decisions before writing SQL. The new field names prevent a purely mechanical copy of the nursery query, and the quantity band adds another required intersection. A record satisfying time, material, and missingness can still fail because its tray count is outside the band.

The tab policy remains deliberately narrow. A10 has a tab-only pickup code, so it is nonmissing under this contract despite looking blank in a grid. A12 contains a nonbreaking space and is also nonmissing. A03 contains ordinary spaces and is missing. State=Packed does not imply a valid pickup code, just as inspection_status=Checked did not imply a nursery bay was assigned. Do not infer cross-field business guarantees that the source does not supply.

Submit an evidence package that distinguishes this independent report from G1–G3: separate SQL, source name, exact output columns, boundary decisions, and stated assumptions. Use the read-only runner and record its engine version. Review the complete result, not only a screenshot. The final learning outcome is a filter whose inclusion decisions can be explained and reproduced, not a query that merely returns a plausible-looking number of rows.

E14-X1 Deliver the independent allocation filter

Write the materials-allocation query from this unit without an answer template. Predict all twelve IDs and give specific reasons for A01,A04,A06,A08,A10,A11,A12. Submit the output together with your three nursery extracts and their grid.

Submit: Independent SQL, complete five-column output, twelve-ID decision record, and final reproducible package.

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. Which end condition is appropriate for the supplied October timestamp interval?

A. received_at < '2026-11-01 00:00'
B. received_at = '2026-10'
C. received_at <= '2026-10-31'
D. received_at > '2026-11-01 00:00'

2. Which expression directly tests SQL NULL?

A. assignee = NULL
B. assignee IS NULL
C. assignee = 'NULL'
D. assignee = 0

3. Why parenthesize two missingness alternatives before adding date conditions?

A. To sort NULL before other values.
B. To convert every whitespace character into NULL.
C. To keep both alternatives subject to the date and category restrictions.
D. To make the database writable.

4. A tab-only assignee looks blank. Under this exercise contract, how is it classified?

A. Missing, because every invisible character is equivalent.
B. NULL, because tabs cannot be stored in text.
C. Invalid SQL, because strings cannot contain tabs.
D. Nonmissing, because only NULL, empty text and ASCII spaces count as missing.

5. Why can direct timestamp text comparisons work in this dataset?

A. SQLite automatically recognizes every text date format.
B. Text comparisons ignore the year component.
C. The table has fewer than twenty rows.
D. All values share a normalized, chronologically sortable format and timezone.

6. Which rule describes the outside of a half-open interval [start,end)?

A. Values below start or at least end.
B. Values at least start and below end.
C. Only values below start.
D. Only values strictly above end.

7. What is wrong with validating only the number of returned rows?

A. Counts are unavailable for filtered queries.

B. Row counts always change stored data.

C. A query cannot return duplicate category values.

D. An incorrect inclusion and exclusion can offset each other.

8. Which category expression clearly selects either permitted category?

A. category = 'Hardware' AND category = 'Software'

B. category IN ('Hardware','Software')

C. category = 'Hardware,Software'

D. category IS NULL

9. The brief does not mention status. Should Closed records be excluded?

A. No; adding that filter changes the specified population.

B. Yes; service requests are always open.

C. Yes; Closed text is equivalent to NULL.

D. Only if excluding them makes the count smaller.

10. Which validation method is most independent?

A. Copy the same WHERE clause into another query and trust agreement.

B. Rerun the query without reading any results.

C. Delete boundary records before running the query.

D. Predict named edge-case membership by inspecting the supplied facts, then compare query output.

11. The receiving week is [2026-11-09 00:00:00, 2026-11-16 00:00:00). A Seeds receipt arrives exactly at the second timestamp and has NULL bay_code. Which main extract includes it?

A. G3 only

B. G2 only

C. G1 only

D. Both G1 and G3

12. A diagnostic wants Seeds or Tools with at least 10 packs. Which grouping implements that requirement?

A. item_class='Seeds' OR (item_class='Tools' AND packs>=10)

B. (item_class='Seeds' OR item_class='Tools') AND packs>=10

C. item_class='Seeds' AND item_class='Tools' AND packs>=10

D. item_class IN ('Seeds,Tools') AND packs>=10

13. N17 contains one nonbreaking-space character, U+00A0. Under the supplied missing-code policy, how should it be classified?

A. SQL NULL because it looks blank.

B. An invalid row that must be deleted.

C. Missing because all Unicode whitespace is included.

D. Nonmissing because default ASCII-space trimming does not remove it.

14. A query uses WHERE bay_code = NULL. Why is it not a valid way to find missing markers?

A. It updates every bay code.

B. It matches only empty strings.

C. Equality with NULL does not produce a true missing-marker test; use IS NULL.

D. It trims tabs before comparing.

15. An inside-week Seeds receipt has bay_code = Bay-F and packs = 0. No quantity restriction is part of G2. What should happen?

A. Exclude it because zero always means missing.

B. Include it in G2 under the stated rules.

C. Include it only if inspection_status is Pending.

D. Move it to G3 because it has no packs.

16. Which evidence best checks that a half-open time interval and its outside complement cover the supplied non-NULL timestamps?

A. Both outputs use the same font.

B. Equal total counts for the two outputs.

C. Disjoint complete ID lists whose union contains every source ID.

D. A screenshot showing one middle-of-week row.

17. A filter uses TRIM(bay_code) in WHERE but returns bay_code directly in SELECT. N18 contains spaces around Bay-E. What is expected?

A. The database permanently loses those spaces.

B. N18 is nonmissing and its returned original value retains the spaces.

C. N18 becomes NULL in the result.

D. N18 is excluded because every string with a space is missing.

18. The independent allocation task accepts 4 through 8 trays, inclusive. Which boundary pair is correct?

A. tray_count > 4 AND tray_count < 8

B. tray_count >= 4 AND tray_count < 8

C. tray_count >= 4 AND tray_count <= 8

D. tray_count > 4 AND tray_count <= 8

19. A faulty G1 leaves OR TRIM(bay_code)=empty text outside parentheses. Which record is the best counterexample to a leaked category restriction?

A. An inside-week Seeds record with NULL bay.

B. An inside-week Seeds record with a normal bay code.

C. An inside-week Pots record with empty bay text.

D. An outside-week Tools record with a normal bay code.

20. Why is it wrong to claim G1, G2, and G3 together cover all nursery receipts?

A. The reports have different category and missingness requirements that leave some records outside all three.

B. A filter cannot return fewer rows than its source.

C. ORDER BY duplicates every receipt.

D. Every NULL bay code belongs to every report.

Sources and further reading

SQLite SQL language expressions
https://www.sqlite.org/lang_expr.html

SQLite SELECT documentation
https://www.sqlite.org/lang_select.html

SQLite built-in scalar functions
https://www.sqlite.org/lang_corefunc.html

SQLite URI filenames
https://www.sqlite.org/uri.html