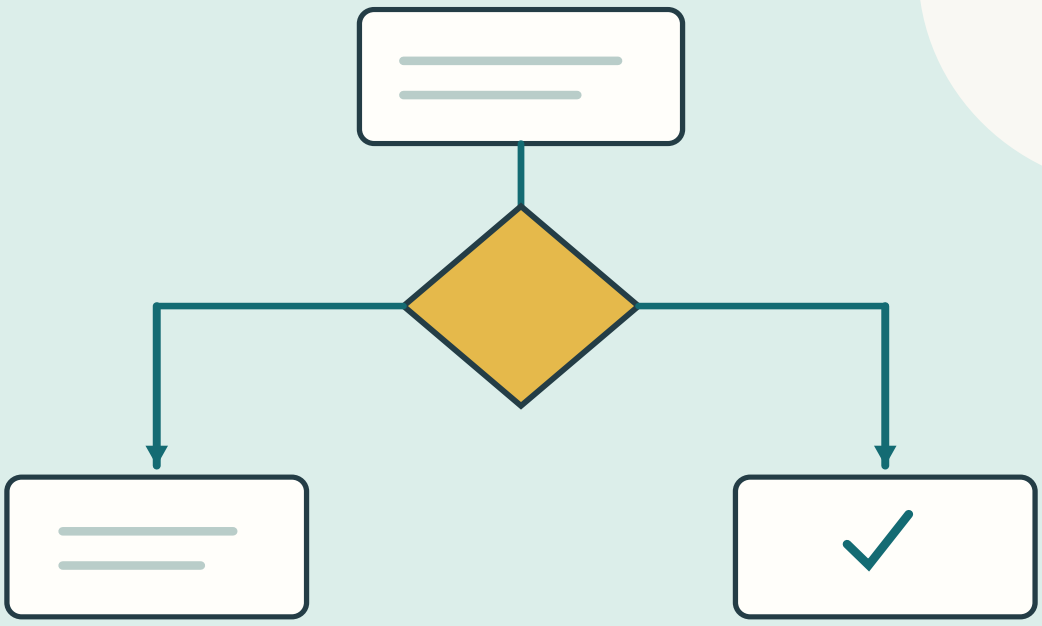
SQL ANALYSIS BANDS

Creating Analysis Bands with SQL CASE



CASE / BANDS

AI & DATA

Creating Analysis Bands with SQL CASE

Learning outcomes

- Translate four explicit processing-time intervals into a deterministic CASE expression.
- Test eight boundary values without gaps or overlapping classifications.
- Reconcile band counts to all eligible records and separate invalid inputs.
- Translate a stated numeric domain into exhaustive, nonoverlapping bands.
- Validate boundary, missing and invalid inputs before reporting band counts.
- Separate stable band ordering from display labels and expose zero-count bands.
- Compare proposed threshold revisions by changed identifiers and transitions.
- Test classifier mutations using independent interval logic and preserve raw precision.

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

Write intervals before writing CASE

A band is a rule assigning a numeric observation to a named interval. It is not a prediction and does not require a model. The intake team defines four intervals measured in minutes. A square bracket at the left means the starting value is included; a parenthesis at the right means the ending value is excluded. Thus a value exactly at a boundary belongs to the band beginning at that boundary, not the band ending there.

Draw the number line in words: from zero up to but not including ten, from ten up to but not including thirty, from thirty up to but not including sixty, then sixty and above. Check that every valid nonnegative value has one place and no valid value has two places. The labels are business descriptions only. “Review” does not prove an error or poor service; it names the team's chosen interval.

The production table is constrained to nonnegative, nonmissing measurements. The separate quality-probe table deliberately violates that domain to show why input validation matters. Do not use a catch-all band as a substitute for a quality policy. First define which values are eligible for analysis, then classify those values. Otherwise a NULL or negative duration can be mislabeled as a legitimate operational case.

Use ordered conditions carefully

A searched CASE expression evaluates conditions in order and returns the result associated with the first condition that succeeds. This supports an ascending-threshold pattern once the valid domain is established: below ten, below thirty, below sixty, otherwise Review. The second branch effectively applies after the first branch has failed. The ordering is therefore part of the logic, not a formatting choice.

An alternative is to write both lower and upper comparisons for each interval. That is more verbose but makes interval boundaries explicit. Either design can be correct if it implements the same contract and handles invalid values separately. Do not mix patterns casually; a broad first condition such as minutes < 60 would capture observations intended for all three lower bands and prevent later branches from being reached.

Use stable, unambiguous labels. If the detail query spells a band Standard and a separate summary query spells it Standard time, reconciliation becomes harder. One useful approach is to classify rows in a named intermediate query and aggregate that derived band field. This avoids duplicating a long CASE expression in several places. The current exercise focuses on the rule and its tests; readability helps reviewers see whether the boundaries actually match the specification.

Test the edges and invalid domain separately

The eight boundary cases are chosen to challenge the intervals rather than merely fill a table. Zero checks the start of the valid domain. Values just below a threshold test the upper edge of a lower band. Exact threshold values test the inclusive lower edge of the next band. A larger value checks the open-ended final interval. Record a predicted band for each named case before executing your classification.

The dataset includes additional ordinary values so you can also test a complete summary. They should not distract from the boundary checks. A classifier can categorize many interior values correctly while mishandling every exact boundary through an incorrect <= operator. That is why the practical criterion requires all eight boundary classifications to match, not merely a high overall percentage.

Run a separate quality query on quality_probes that identifies NULL or negative durations. Keep that result outside the production totals. If you experiment with applying the CASE expression to the probes, explain why the unguarded result would be misleading. A final ELSE branch means “no earlier condition was true,” not “the value is known to be a valid large number.” The input-domain check is what makes that interpretation safe for production measurements.

Reconcile the classification and communicate limits

Every production record must receive exactly one valid band. Count classified detail rows and compare with the production source count. Then sum the four band counts and compare again. These two checks ensure that classification and aggregation preserve the intended population. Also verify that all four labels are present in this fixture and that the operational ordering is Quick, Standard, Extended, Review rather than alphabetical order.

A numerical sort key associated with each band can make the intended presentation order explicit. Keep the label human-readable and the sort rule separate. Do not rely on insertion order or the accidental alphabetical sequence of labels. The report should remain understandable even if a business owner later changes the displayed wording while leaving interval boundaries unchanged.

Submit the detail classification, four-band summary, eight-row boundary test sheet and separate invalid-probe result. Explain the unit, valid domain and exact interval endpoints in a short note. Do not claim the bands are externally validated performance standards; they are supplied synthetic business rules. The central evidence is that your deterministic SQL implements those rules completely, excludes invalid inputs from legitimate bands and preserves all intended observations without dropping or duplicating cases.

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: First-match behavior on a different threshold

Sql

```
WITH demo(x) AS (VALUES (4),(5),(8)) SELECT x,CASE WHEN x<5 THEN 'Below five' ELSE
    'At least five' END AS band FROM demo ORDER BY x;
```

Explanation: The exact boundary value 5 enters the second branch. Replacing < with <= would implement a different interval contract.

Foundation practice

Scenario

A fictional document intake team groups nonnegative processing minutes into four operational bands. The production fixture has twelve valid measurements; a separate probe table contains invalid values for a quality-check rehearsal.

Database file: 3H-AI-0148_practice/practice.sqlite

Schema file: 3H-AI-0148_practice/schema.sql

Runner file: 3H-AI-0148_practice/run_query.py

Readme file: 3H-AI-0148_practice/README.txt

Tables

processing

case id	minutes	test role
T01	0	boundary
T02	9.9	boundary
T03	10	boundary
T04	29.9	boundary
T05	30	boundary
T06	59.9	boundary
T07	60	boundary
T08	90	boundary
T09	4	ordinary
T10	18	ordinary
T11	44	ordinary
T12	75	ordinary

Csv file: processing.csv

quality_probes

probe id	minutes
X01	NULL
X02	-1
X03	10

Csv file: quality_probes.csv

Requirements

Classify processing rows into exactly four bands: Quick [0,10), Standard [10,30), Extended [30,60), Review [60,infinity). Produce case_id, minutes, band in ID order, plus band counts in the specified operational order. Test T01-T08 individually and preserve all twelve processing rows. Separately identify invalid quality_probes values (NULL or negative); do not silently place invalid probes into Review or include them in the production band totals.

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-0148_practice/boundary_test_blank.csv

Foundation tasks

- Exercise 1: Write the four intervals in plain English and predict bands for T01-T08.
- Exercise 2: Identify invalid values in quality_probes and explain why a final ELSE cannot validate the input domain.
- Capstone: Deliver detail and ordered band counts, pass all eight boundary tests and reconcile all twelve production records without including invalid probes.

01 Make a band specification precise enough to test

The fictional Cedar Scanning Studio records how many seconds each completed scanning job spent in a processing queue. The workshop does not ask you to estimate delays or explain their causes. It asks you to translate an explicit numeric policy into reproducible analysis bands. A band assigns one observation to a named interval; it does not change the observed value or establish whether a person performed well.

The four labels are Brief, Steady, Long and Review. Brief includes zero and stops before 120 seconds. Steady starts at 120 and stops before 300. Long starts at 300 and stops before 900. Review starts at 900 and has no upper boundary in this exercise. These are supplied fictional analysis rules, not industry standards, service commitments or instructions to contact anybody.

State the unit and domain before writing CASE. The main jobs table contains twenty-two nonmissing nonnegative whole-second observations. Its schema enforces the nonmissing and nonnegative conditions. All jobs and both shifts belong in the production classification. The intake_probes and precision_probes tables serve separate tests; neither should be added to the production population merely because its column has the same name.

An interval with an included lower endpoint and excluded upper endpoint is half-open. The convention assigns a value exactly at a shared threshold to the interval that begins there. Thus the endpoint has one owner. A specification that says short up to 120 and medium from

120 without clarifying inclusion is incomplete: both phrases could include 120. Write comparisons that remove that ambiguity before choosing labels or colors.

The detail report must contain job_id, raw elapsed_seconds and band, ordered by job_id. The summary must show band_rank, band and job_count in the operational order 1 through 4. Every valid production job receives exactly one band. Preserve the raw seconds so a reviewer can check a boundary independently instead of trusting a label with no supporting measurement. The original practice uses minutes and different labels; keep that separate rule set intact.

Rank	Band	Included lower bound	Excluded upper bound
1	Brief	0	120
2	Steady	120	300
3	Long	300	900
4	Review	900	No upper bound

E01-X1 State the production classification contract

Write the unit, valid domain, all four intervals, exact detail and summary fields, population count and explicit rule for 120, 300 and 900. Identify which tables are not production jobs.

Submit: A complete rule contract without SQL.

02 Build an independent boundary ledger before coding

Boundary tests are strongest when their expected answers come from the written intervals rather than from the SQL being tested. The jobs fixture deliberately includes exact thresholds and nearby integer values. Zero checks the start of the valid domain. Values 119, 120 and 121 bracket the first change; 299, 300 and 301 bracket the second; 899, 900 and 901 bracket the third. The value 1200 tests that the final interval remains open-ended.

Create a small ledger with identifier, raw value, expected interval and the reason. For 299, the reason is at least 120 but below 300. For 300, the reason changes: it is at least 300 and below 900. Stating the reason prevents an expected-answer sheet from becoming a set of unexplained labels that can accidentally be copied from a faulty query output.

Interior observations also matter. Values such as 240 and 600 show that the bands are not merely a list of special thresholds. Repeated values test consistency: J06 and J20 both equal 300, so both must receive Long. J09 and J21 both equal 900, so both must receive Review. A rule should not depend on insertion order, shift or identifier unless those fields are explicitly part of the contract.

A boundary ledger need not include every mathematically possible real number. It should target the points where an implementation choice can change membership, plus the stated invalid and empty states. The whole-second production domain makes one-second neighbors

natural here. A later precision exercise uses fractional seconds, where rounding creates a separate risk. Keep the measurement resolution and the chosen test distance explicit.

Do not measure success only by the percentage of ordinary rows classified correctly. An incorrect <= comparison can classify many interior values correctly while assigning all exact thresholds to the wrong band. Since boundaries define the rule, a single unresolved threshold error is a substantive defect, not a cosmetic difference. The ledger should make that failure visible even when the overall row count still reconciles.

Probe value	Boundary question
0	Is the domain start included?
119	Does the lower band stop before 120?
120	Does the next band own equality?
121	Does the new interval extend above its start?

E02-X1 Predict the ten-row threshold spine

Using J01 through J10, list job_id, elapsed_seconds and expected band without executing CASE. Add the expected labels for J13=1200 and J20=300. Explain why J06 and J20 must agree.

Submit: Twelve named predictions with boundary reasons.

03 Use searched CASE with deliberate first-match order

A searched CASE checks Boolean conditions in their written order and uses the result attached to the first true condition. An ascending threshold design can exploit that behavior after the valid domain is established. A value below 120 is Brief. If it is not below 120 but is below 300, it is Steady. If neither earlier condition succeeds but it is below 900, it is Long. The remaining valid values are Review.

The lower bounds are implicit in the earlier failed branches. For example, the Steady branch written as seconds < 300 also tests true for 60 in isolation. In the complete CASE, 60 has already been assigned Brief, so the later branch is not selected. The ordered expression implements disjoint output even though individual threshold predicates overlap. This is intentional and should be explained rather than mistaken for an inherently invalid design.

The order is part of the meaning. Putting seconds < 900 first would capture 0, 119, 120, 299 and many other values before their more specific branches. The query would still execute, return one label per row and preserve the source count. Those superficial checks cannot detect the semantic error. The independent boundary ledger is what exposes the misplaced broad condition.

Use an ELSE only after understanding what can reach it. In the constrained jobs table, every remaining value is nonnegative, nonmissing and not below 900, so Review is justified. The

same expression is not safe for the unconstrained intake probes. A missing value does not make any numeric threshold condition true and would fall into ELSE, while a negative value would satisfy the first branch. Domain validation is a precondition, not something CASE automatically supplies.

A readable expression puts one condition and result on each line, with consistent labels and an explicit alias. Preserve complete runnable SQL rather than replacing repeated details with ellipses. Reviewers should be able to execute the saved statement and see exactly which conditions were tested. The worked miniature demonstrates first-match order; your practical query must classify the entire production table.

```
WITH sample(seconds) AS (VALUES (119),(120),(300),(900))
SELECT seconds,
       CASE WHEN seconds < 120 THEN 'Brief'
            WHEN seconds < 300 THEN 'Steady'
            WHEN seconds < 900 THEN 'Long'
            ELSE 'Review' END AS band
FROM sample ORDER BY seconds;
```

E03-X1 Classify the entire production population

Write the ascending-threshold CASE for jobs. Return job_id, elapsed_seconds and band in ID order. Verify all 22 rows and the repeated threshold values, without filtering shifts or zero values.

Submit: Runnable SQL and all 22 classified rows.

04 Compare ordered and explicit interval designs

Another valid design writes both the lower and upper bounds for each finite interval. Brief requires seconds >= 0 AND seconds < 120. Steady requires seconds >= 120 AND seconds < 300, and Long requires seconds >= 300 AND seconds < 900. Review requires seconds >= 900. These predicates describe disjoint intervals directly, so their mathematical meaning is less dependent on a reader remembering earlier failed branches.

The explicit design is not automatically better. It repeats threshold values, creating more places where a revision can be applied inconsistently. If one branch ends at 360 while the next still begins at 300, the predicates overlap. If one ends at 300 and the next begins at 360, values between them fall into a gap. A named rule specification and tests are still needed whichever expression looks clearer.

BETWEEN includes both endpoints in SQL. Using BETWEEN 120 AND 300 for Steady would include 300, even though the current policy assigns 300 to Long. If another branch also includes 300, the first-match behavior may hide the overlap by returning one label. The fact that CASE returns only one output does not prove that the underlying interval definitions have exactly one membership for every valid value.

You can test the two designs against the same immutable input and compare results by job ID and band. An empty difference supports equivalence on this fixture, not a theorem about every possible input or any later edited threshold. The independent interval ledger remains

the source of expected meaning. If both implementations share a mistaken <= endpoint, they can agree on the same wrong answer.

For review, choose one canonical implementation for reporting and use the second as a diagnostic. Do not let the detail report use one threshold set while the summary uses another. A named derived classification or stable numeric rank helps avoid that drift. This course uses ordinary query expressions for readability without promising a particular physical execution strategy or permanent stored rule table.

Expression for Steady	Includes 120?	Includes 300?	Matches policy?
x >= 120 AND x < 300	Yes	No	Yes
x BETWEEN 120 AND 300	Yes	Yes	No
x > 120 AND x < 300	No	No	No

E04-X1 Verify an explicit interval alternative

Write a CASE using explicit lower and upper comparisons and an explicit >=900 branch. Use ELSE NULL as a diagnostic fallback. Return all job classifications and compare with E03. Identify the two endpoint defects in the table.

Submit: Full equivalent detail SQL plus exact endpoint explanations.

05 Keep missing and invalid inputs outside legitimate bands

The intake_probes table deliberately lacks the production table's domain constraints. It contains missing values and negative values alongside legitimate thresholds. Treat it as a quality rehearsal, not as extra production observations. The classification policy for these probes is explicit: NULL produces quality_state MISSING, negative produces INVALID_NEGATIVE, and every nonnegative numeric value produces VALID. Only VALID observations receive one of the four ordinary bands.

A useful output retains probe_id and raw elapsed_seconds, then provides quality_state and band separately. For invalid or missing input, band should be NULL. This preserves the distinction between a failed domain check and a legitimate long duration. Putting INVALID into the same field as ordinary bands can be workable with a clear contract, but a separate quality field makes the denominator and downstream filtering easier to review here.

Guard the invalid states before the ordinary threshold branches. A negative duration satisfies seconds < 120, so putting the negative guard after that condition would be too late. A NULL input needs IS NULL, not = NULL. Numeric comparisons with NULL do not yield true, and a simple CASE with a WHEN NULL equality branch is not a substitute for an explicit missingness test.

The guard does not change the raw value. It adds a derived interpretation under the supplied domain policy. Do not replace a missing duration with zero before classification. Zero is a

valid observed duration assigned Brief; it is not a neutral code for unknown. Similarly, taking the absolute value of a negative duration would hide an invalid input and invent a legitimate measurement.

Quality reporting should reconcile its own population. There are twelve intake probes, not twenty-two jobs. Count valid, missing and invalid-negative states so that their sum equals twelve, then reconcile only valid probe bands to the valid-probe count. Keep this rehearsal separate from the production summary. A report that mixes the tables can preserve a grand count while using the wrong denominator for operational analysis.

E05-X1 Classify guarded intake probes

Return probe_id, elapsed_seconds, quality_state and band for all intake probes in ID order. Use MISSING, INVALID_NEGATIVE and VALID; keep band NULL on the first two states. Reconcile all three state counts.

Submit: Twelve exact rows and state-count evidence.

06 Prove coverage instead of assuming one CASE result is enough

A CASE expression returns one scalar result per input row, but that property alone does not establish a complete and exclusive interval specification. If no branch matches and no ELSE is provided, the result is NULL. If several branches would be true, the first wins and can hide overlap. Test the interval predicates themselves when reviewing coverage, not only the final label column.

For a valid observation, calculate four membership indicators using the explicit disjoint interval definitions. Each indicator is one when its interval contains the value and zero otherwise. Add them. The expected total is exactly one for every valid job. A zero indicates a gap; a value greater than one indicates overlapping definitions. This diagnostic is separate from the compact ordered CASE, whose individual upper-bound tests intentionally overlap.

The distinction matters when a rule is revised. A careless editor could change Steady's upper bound to 360 without changing Long's lower bound from 300. Values 300 and 301 then satisfy both explicit intervals. CASE still gives one label because it selects the first true branch, but the membership-count test reveals two memberships. The specification has become inconsistent even if the output looks tidy.

Use domain guards when checking coverage on nullable probes. Comparisons involving NULL can themselves be NULL, and adding raw comparison results may propagate that unknown state. The production domain is already constrained here, so the worked diagnostic is straightforward. For mixed inputs, validate first and run the interval-coverage test only on valid rows or explicitly encode each indicator with CASE.

Coverage does not require every band to contain observations. An interval can be valid and empty in a particular population. The diagnostic checks whether each supplied valid observation belongs somewhere exactly once, not whether all labels appear. A later quiet-job

example shows how to display zero-count bands without inventing observations. Keep rule completeness, observed coverage and presentation completeness as separate questions.

```
SELECT job_id,
       (elapsed_seconds >= 0 AND elapsed_seconds < 120)
     + (elapsed_seconds >= 120 AND elapsed_seconds < 300)
     + (elapsed_seconds >= 300 AND elapsed_seconds < 900)
     + (elapsed_seconds >= 900) AS memberships
FROM jobs
ORDER BY job_id;
```

E06-X1 Locate a proposed overlap

For jobs, change only the explicit Steady upper bound to 360 while leaving Long starting at 300. Return job_id, elapsed_seconds and memberships for rows whose membership count is not 1. State why a first-match CASE alone would conceal the overlap.

Submit: Exact failing IDs and runnable coverage diagnostic.

07 Summarize a classification without changing its population

The summary groups already-classified jobs, so every production job contributes once to exactly one band count. Use a derived band rank to connect the classification to the band_order table. The rank describes interval order, while the label is human-readable presentation. Keeping the rank separate prevents a later wording change from silently altering the sort order or forcing a reviewer to infer numeric meaning from alphabetical text.

The main fixture has five Brief, five Steady, seven Long and five Review jobs. Their counts sum to twenty-two. These numbers are not threshold values or average durations. Label job_count explicitly so a reader cannot mistake the measure for seconds. The raw detail remains the place to inspect how individual durations contributed to each group.

Counting classified rows before grouping and summing group counts after grouping checks two stages of population preservation. Neither check establishes that the labels are correct: the widest-first CASE would still produce twenty-two classified rows and a summary summing to twenty-two. Combine count conservation with boundary correctness and valid-domain tests rather than choosing one preferred check as a universal proof.

A summary should derive from one canonical classification. Copying CASE separately into a detail query and a summary query invites drift when a threshold or label is revised. A named intermediate query is useful here because it gives the derived rank a clear role. That is a readability and consistency choice, not a promise that the engine stores an intermediate table or executes a particular physical sequence.

Presentation order must be explicit. Alphabetical labels would put Brief, Long, Review and Steady in a misleading sequence. ORDER BY band_rank gives the requested interval progression. Sorting by count answers another question, which band is most common, and may be useful in a separate display. It must not replace the required operational-order summary without being labeled as a different view.

Rank	Band	Production jobs
1	Brief	5
2	Steady	5
3	Long	7
4	Review	5

Count the selected state, not every nonmissing indicator

CASE is also useful inside an aggregate when a report needs one selected band count beside a total. Be precise about the aggregate receiving its result. COUNT(expression) counts non-NULL results, including numeric zero. Therefore COUNT(CASE WHEN elapsed_seconds >= 900 THEN 1 ELSE 0 END) counts every production job, not only Review jobs. The zeros are ordinary nonmissing values, so they contribute to COUNT just like the ones.

Two correct alternatives express different mechanisms. COUNT(CASE WHEN elapsed_seconds >= 900 THEN 1 END) leaves the false result NULL because ELSE is omitted, so only qualifying jobs contribute to the count. SUM(CASE WHEN elapsed_seconds >= 900 THEN 1 ELSE 0 END) instead adds one for each qualifying job and zero for each other job. Both return five on the nonempty production fixture, but the first counts nonmissing markers while the second adds numeric indicators.

The distinction becomes visible on an empty population. Both COUNT expressions return zero when there are no rows to inspect. The SUM expression returns NULL because it has no numeric inputs, even though its CASE contains ELSE zero. ELSE is evaluated for an existing row whose condition is not true; it does not manufacture a row when the population is empty. If the display requires zero, apply an explicit documented COALESCE to the aggregate result and keep the source row count available.

This diagnostic complements grouped band counts. A single-row dashboard can look cleaner than a four-row grouped report, but compact layout must not hide a change in counting semantics. Review the false branch together with the aggregate function. An indicator called is_review does not become a count merely because its values are zero and one. Write the intended measure in words, then check whether the chosen aggregate counts markers or sums them. The same habit prevents a plausible twenty-two from being mislabeled as the Review population.

E07-X1 Produce the ordered production summary

Classify jobs once by numeric band_rank, attach band_order, and return band_rank, band and job_count. Reconcile four counts to 22 and explain why this does not replace boundary tests.

Submit: Four exact summary rows and runnable SQL.

E07-X2 Diagnose a conditional-count trap

On jobs, compare COUNT with a CASE returning 1 or 0, COUNT with CASE returning 1 or NULL, and SUM with CASE returning 1 or 0 for elapsed_seconds >= 900. Include total_jobs. Repeat on an empty jobs population. Explain the four outputs in each row.

Submit: Two complete SQL queries and exact aggregate rows.

08 Show zero-count bands without manufacturing jobs

The quiet_jobs table contains only three observations: zero, sixty and one hundred nineteen seconds. All fall in Brief. A grouped summary over observed classifications alone returns just Brief with a count of three. That result is valid as an observed-category summary, but the requested quiet-period display must show all four policy bands, including those with no observations.

Start that display from the four-row band_order table, then attach classified quiet jobs by rank using a LEFT JOIN. This preserves the category domain independently of whether any job occupies it. Count a nonmissing job identifier on the attached side, not COUNT(*). An empty category still contributes a NULL-extended joined row, so COUNT(*) would misleadingly report one job in each empty band.

The correct four-band output has Brief three and the other bands zero. Those zero counts are not new source records. They describe a category that exists in the reporting scheme but has no matched observations in the chosen population. This is different from a missing job duration, which is an existing source row whose measurement is unknown. Conflating those states can lead to either invented activity or lost quality exceptions.

If the job population is entirely empty, the same category-driven display should show four zeros. The detail result should be empty, and the source count should be zero. SUM over no observed durations remains NULL unless the presentation contract explicitly requests a zero display. Do not confuse filling a missing summary count with replacing missing source measurements. The former completes a reporting domain; the latter changes evidence.

A useful acceptance test compares the sum of displayed counts with the quiet-job source count. It also asserts that exactly four distinct ranks appear in the requested order. These checks test both population conservation and presentation completeness. Neither requires every band to have a positive count, and neither justifies deleting a category because it happens to be unused in one small snapshot.

E08-X1 Create a full quiet-period band display

Classify quiet_jobs and LEFT JOIN from band_order. Return rank, label, count with all four bands. Also run the same display with an empty quiet-job population. Explain why COUNT(job_id), not COUNT(*), is required.

Submit: The (3, 0, 0, 0) and (0, 0, 0, 0) displays with complete SQL.

09 Preserve measurement precision at the boundary

The precision_probes table uses fractional seconds to test a different failure from an incorrect endpoint operator. Values 119.9, 120.0 and 120.1 surround the first threshold. Equivalent triplets surround 300 and 900. The policy classifies raw elapsed seconds. Rounding to a whole second before CASE changes which side of the threshold a value occupies, even if the displayed number looks more convenient.

For example, 119.9 is below 120 and belongs in Brief. Rounding it to zero decimal places yields 120, which belongs in Steady. The classification change is caused by the preprocessing decision, not by a different interval policy. A report that displays rounded seconds but labels from raw seconds may show 120 beside Brief; that can be acceptable only if the display explains the rounding and preserves access to the original measurement.

Unit conversion can create another boundary risk. A value measured in seconds must be compared with second thresholds or consistently converted to the chosen unit. Comparing 300 seconds directly to the original practice's 30-minute threshold is not a clever reuse of SQL; it applies a different rule by a factor of sixty. Keep field names and output labels explicit enough to reveal a unit mismatch during review.

For production whole-second jobs, the integer values and integer thresholds make the intended comparisons direct. The fractional probes are educational boundary tests, not a promise about arbitrary floating-point precision. Independent expected classification can use the decimal text recorded in the CSV to avoid deriving the expectation from a rounded display. Do not claim that all possible decimal measurements are stored exactly by a general REAL column.

Choose a presentation policy after classification, not by quietly altering the observed data. If a business owner later requires classification on rounded minutes, that is a new rule with new boundary behavior and should be tested as such. The current task uses raw seconds throughout. Retaining both raw and displayed values when necessary makes this difference visible instead of hiding it in an export formatter.

```
SELECT probe_id, elapsed_seconds,
       ROUND(elapsed_seconds, 0) AS displayed_seconds
FROM precision_probes
WHERE probe_id IN ('F01','F04','F07')
ORDER BY probe_id;
```

E09-X1 Locate rounding-induced reclassification

Classify precision_probes from raw values and from ROUND(elapsed_seconds, 0) using the same thresholds. Return probe_id, raw_seconds, raw_band, rounded_band only when the labels differ. Also provide the complete raw probe classification.

Submit: Three changed rows, nine raw rows and runnable SQL.

10 Compare a threshold proposal without silently adopting it

A proposed rule change extends Steady from below 300 to below 360 seconds. Long would then begin at 360 and still stop before 900. Brief and Review remain unchanged. This is an explicit comparison exercise, not an approved replacement for the current production rule. Keep old and proposed labels in separate columns so the effect can be reviewed without overwriting the established result.

The proposal changes both sides of the shared boundary. Updating only one explicit interval would create the overlap or gap tested earlier. In an ascending-threshold CASE, replace the second upper threshold consistently and retain ascending order. Record the proposed intervals in words as well as code; otherwise a reviewer cannot distinguish an intentional policy change from an accidental edit.

The affected region is at least 300 and below 360. Values below 300 already belong in Steady, and values at or above 360 remain Long unless they reach Review. In the supplied production snapshot, J06, J07 and J20 occupy the affected region. A new probe exactly at 360 should remain Long under the proposal, while 359 should be Steady. These extra probes test the revised boundary, not only the existing observed jobs.

A transition report can show old_band, proposed_band and job_count. Most rows stay on the diagonal because their labels do not change. The off-diagonal Long-to-Steady transition describes reclassification under a rule proposal, not improved processing performance. No elapsed seconds changed. Calling the new distribution faster would confuse a changed ruler with a changed measurement.

Preserve the old summary, proposed summary and changed-ID list together. The total number of jobs must remain twenty-two, and every raw duration must remain the same. A revised classification can legitimately move counts between bands while conserving the source population. That is a different situation from dropping long jobs to make the report look better, which would change both the population and the interpretation.

E10-X1 Quantify the proposed 360-second boundary

Return changed job IDs with raw seconds, old band and proposed band. Produce both ordered summaries, and classify proposed-rule probes 359 and 360. State whether any performance improvement has been demonstrated.

Submit: Three changed rows, two complete summaries and two boundary results.

11 Use denominators that match the question

Band counts are often converted into shares, but a percentage is meaningful only when its denominator is explicit. For the main production table, all twenty-two jobs are valid and in scope. Review contains five jobs, so its share is five divided by twenty-two. For the intake probe rehearsal, only eight observations are valid, and two of those receive Review. The valid-probe Review share is therefore two divided by eight, not two divided by twelve.

The twelve-probe denominator answers a different question: what fraction of all imported probe records are classified Review? That may be a useful reporting measure if clearly labeled, but it mixes ordinary band membership with missing and invalid states in the denominator. It should not be presented as the distribution of valid durations. Report the invalid counts alongside the distribution so the reader can see what was excluded and why.

Use a real-valued numerator or denominator when a fractional result is required. In SQLite, integer division of two integers can discard the fractional part. Multiplying the count by 100.0 before dividing expresses the percentage calculation clearly. Round only the final display, after classification and aggregation. That reporting operation does not alter raw durations or band assignments.

Subgroup denominators need the same care. A Day-only percentage must use the Day population, not all jobs, unless the intended question is the subgroup's contribution to the whole. A comparison can be mathematically correct but mislabeled. Write the numerator and denominator in words before computing them: Review jobs divided by all valid jobs in the selected population is a defensible contract.

For an empty valid population, the rate is undefined rather than evidence of perfect performance. A safe expression can use NULLIF(denominator, 0) to avoid dividing by zero and preserve NULL as not defined. If the presentation later displays a dash or not applicable, document that label. A zero percentage means a nonempty measured population with no numerator events; it is not the same as having no measurements at all.

```
SELECT 5 / 22 AS integer_ratio,
       ROUND(100.0 * 5 / 22, 2) AS review_percent,
       100.0 * 0 / NULLIF(0, 0) AS empty_percent;
```

E11-X1 Report valid-probe shares and quality coverage

Using intake_probes, give valid, missing and negative counts. Compute Review among valid probes and Review among all probes to two decimals, label both denominators and give the empty-valid-population rate.

Submit: Counts 8, 2, 2; percentages 25.00 and 16.67; undefined empty rate, with SQL.

12 Challenge plausible classifiers with targeted mutations

A classifier can fail while remaining syntactically valid and preserving every row. The most informative tests change one logical choice at a time and verify that the expected evidence detects it. Replacing < with <= at the first threshold moves exact 120 into Brief. Putting the widest threshold first moves many short observations into Long. Omitting the NULL guard on intake probes sends missing values into Review.

Each mutation should have a named witness. J03 witnesses the first endpoint defect. J01 witnesses the widest-first defect because its zero seconds cannot legitimately be Long. P01 witnesses the missing guard defect. A test suite without these witness values may pass an incorrect implementation simply because its inputs never reach the flawed condition. Testing many random interior values does not guarantee coverage of a shared boundary.

Some checks remain green under a mutation. All twenty-two production jobs still receive a label after the <= edit. The band counts still sum to twenty-two. A format check still sees the expected columns. This does not make those controls useless; it shows their limits. Combine structural checks, independent expected labels, edge probes and population reconciliation so that each guards a different failure mode.

Do not repair the frozen source to make an incorrect query pass. Changing J03 from 120 to 119 would hide the endpoint defect and destroy the test's purpose. Save mutated SQL separately and compare its output with the canonical expected ledger. Where a mutation is supposed to fail, record the concrete discrepancy rather than merely writing rejected. This makes the control behavior reproducible for another reviewer.

A test that rejects every candidate is also defective. The correct guarded classifier must pass the same review process, including the missing and negative cases. Pair rejection tests with a known-correct control so that strictness does not turn into indiscriminate failure. The objective is to distinguish correct from incorrect implementations under the stated contract, not to maximize the number of red flags.

Mutation	Named witness	Wrong result
First threshold <=120	J03=120	Brief instead of Steady
<900 first	J01=0	Long instead of Brief
No NULL guard	P01=NULL	Review instead of no band

E12-X1 Record three falsifying witnesses

Run the <=120 production mutation and identify changed IDs. Run the widest-first CASE on J01. Run the unguarded CASE on P01. Give the expected and actual labels for each and name a coarse check that still passes.

Submit: Concrete SQL and three mutation findings.

13 Transfer interval logic to an hours-based hold report

The holds table is an independent fictional library pickup analysis. Its measurement is age_hours, not elapsed_seconds. The supplied rule defines Fresh from zero up to six hours, Waiting from six up to twenty-four, Aged from twenty-four up to seventy-two, and Review from seventy-two onward. Missing and negative ages remain quality exceptions with no ordinary band. The labels describe recorded age intervals only.

This transfer requires more than changing table names. Units, thresholds, field names and population size all differ. The source contains twelve hold records, including ten valid ages, one missing age and one negative age. Some values are fractional so that 5.9, 23.9 and 71.9 test the upper edges directly. Exact 6, 24 and 72 test ownership of the next interval's lower edge.

No creation or observation timestamps are supplied. Treat age_hours as the given measurement rather than reconstructing it from the current date. You cannot infer which calendar day a hold began, whether it is overdue under an external policy or whether a customer should be contacted. A band label named Review is not a communication authorization or a decision to cancel a hold.

Produce one detail row per hold, retaining raw age, quality_state and band. Summarize only the ten valid holds in operational order. Keep quality counts beside that summary so the missing and negative inputs remain visible rather than disappearing silently. The valid-band count sum should equal ten, and valid plus missing plus negative should equal twelve.

To review the result independently, place each raw age into the written interval before comparing with SQL. The same discipline used for the production jobs applies, but the expected numbers must be recalculated from this new source. Copying the earlier summary counts would be a recognizable template error. Transfer competence means retaining the reasoning pattern while rebuilding the concrete contract and evidence for the new case.

E13-X1 Build the independent hold-age report

Write guarded CASE detail for holds with hold_id, age_hours, quality_state, band. Produce the ordered valid-band summary and separate quality counts. Test the exact 6, 24, 72 boundaries and state the limits on dates or outreach.

Submit: All 12 detail rows, four valid-band counts, quality counts and full SQL.

14 Audit subgroup summaries without redefining the bands

A studio reviewer asks for band counts separately for Day and Night shifts. This changes the summary grain to one shift-and-band combination; it does not change the band thresholds or the source eligibility rule. Every production job still receives the same band as before. A job's shift is a grouping dimension, not an instruction to apply a different classifier.

The Day population contains twelve jobs and the Night population ten. For each shift, count jobs by band rank. Then check that each shift's band counts sum to its source count and that the two shift totals sum to twenty-two. Finally, add the two counts for each band and compare with the overall summary. This two-direction reconciliation tests both the subgroup partition and the band distribution.

A common mistake is to use the total production denominator for every shift percentage. Another is to classify only one shift correctly and reuse those counts for the other. The concrete source IDs let a reviewer inspect the groups: Day includes J01 through J12, while Night includes J13 through J22. Both shifts include zero and exact-threshold values, so neither is exempt from the same boundary logic.

Do not infer a causal shift effect from these small synthetic counts. The exercise establishes descriptive distributions for supplied records. It does not control for job type, staffing, workload or any other cause of processing time. A larger Review count in one shift would not

by itself establish that the shift caused longer waits. Keep the conclusion at the level the supplied observations support.

When a subgroup has no rows in a band, decide whether the display contract requires an explicit zero. The general category-driven approach from quiet_jobs can be extended to a declared shift-by-band domain. For this fixture all eight combinations are observed, so an ordinary grouped result is sufficient for the requested output. State that fact instead of assuming every future snapshot will have the same observed combinations.

E14-X1 Reconcile shift-by-band counts in both directions

Return shift, band_rank, band, job_count in shift then rank order. Reconcile Day and Night totals and each band total to the overall production summary. Include one limitation on causal interpretation.

Submit: Eight exact grouped rows and two-direction arithmetic.

15 Deliver a classifier that another reviewer can challenge

The completed band-analysis pack should preserve the policy, raw observations, SQL and checks together. A chart of band counts is not enough to verify endpoint ownership, missing-input handling or a revised threshold. Save the detail classification and ordered summary, the independent boundary ledger, the quality-probe results and the proposed-rule comparison as distinct artifacts with unambiguous population names.

The acceptance note should distinguish current and hypothetical rules. The current production counts are five Brief, five Steady, seven Long and five Review. The proposed 360 boundary produces five, eight, four and five without changing any raw duration. The note must not claim performance improvement from that relabeling. The same distinction protects downstream readers who may see only a summary chart rather than the query definitions.

Document three kinds of evidence. Semantic evidence shows that intervals, guards and exact thresholds match the contract. Population evidence shows that detail rows, band counts and quality states conserve the intended records. Reproducibility evidence shows which local snapshot and runnable query produced each output, with original files unchanged. A pass in one category does not compensate for an unresolved defect in another.

The supplied runner opens the fictional database read-only and permits result-producing queries. Use it from the course folder with a separately saved SQL file. No installation beyond Python's standard SQLite module is needed. Keeping source files unchanged preserves the evidence; it does not prevent a wrong CASE from running. The targeted mutation results are therefore part of the final review rather than optional debugging notes.

A final conclusion should be specific enough to reject a near miss. All exact thresholds must be assigned correctly, no invalid probe may receive an ordinary band, all production jobs must remain present, and the ordered counts must reconcile. If a single threshold is still wrong, request revision even when the overall chart looks plausible. If every stated check

passes, accept the supplied-snapshot implementation without claiming universal performance standards or authorization for downstream operational actions.

E15-X1 Assemble the final band-analysis evidence pack

Submit current production detail and summary, threshold ledger, guarded intake results, zero-band display, precision comparison, proposed 360 changes and shift reconciliation. Write an acceptance note naming the expected counts and one failure caught by each test type.

Submit: Complete SQL/results from the earlier exercises plus a bounded verdict.

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 minute rule assigns Standard to values from 10 inclusive to 30 exclusive. Which SQL condition implements it?

A. minutes >= 10 AND minutes < 30

B. minutes > 10 AND minutes <= 30

C. minutes BETWEEN 10 AND 30

D. minutes > 10 AND minutes < 30

2. In a searched CASE, a value satisfies two WHEN conditions. Which branch determines its returned label?

A. The branch with the largest numeric threshold.

B. The last true branch.

C. The first true branch.

D. A random true branch selected during grouping.

3. A valid production CASE puts minutes <60 before minutes <10 and minutes <30. What defect results for a 5-minute record?

A. It is rejected as outside the nonnegative domain.

B. It receives the label of the final ELSE.

C. It receives two output rows.

D. It is captured by the broad first condition.

4. What must a complete, nonoverlapping band specification guarantee for every valid input?

A. Exactly one interval contains that input.

B. Every band has an equal observed count.

C. Every interval has the same width.

D. Every input is exactly a threshold value.

5. Why do tests include 10, 30 and 60 exactly in the original minute case?

A. To prove that all raw measurements are whole numbers.

B. To make ordinary-value tests unnecessary.

C. To detect wrong inclusive or exclusive endpoint operators.

D. To establish that the labels are external service standards.

6. An unguarded ascending CASE ends with ELSE Review. What can happen when minutes is NULL?

A. CASE automatically flags an invalid measurement.

B. NULL is treated as zero and becomes Quick.

C. No numeric condition is true, so Review can be returned.

D. The row is omitted from SELECT.

7. The original production table has 12 valid rows and a separate quality_probes table. Which population belongs in the production band summary?

A. All rows in both tables.

B. Only the eight boundary production rows.

C. Only production values below 60.

D. All 12 production rows, excluding the separate probes.

8. Which pair of counts checks classification conservation for the original production population?

A. Number of labels and the largest minute value.

B. Classified detail rows and the sum of band counts, each compared with 12.

C. Number of probes and the number of thresholds.

D. Count of non-NULL labels only, with no source comparison.

9. The requested order is Quick, Standard, Extended, Review. Which design keeps that order independent of label spelling?

A. ORDER BY band alphabetically.

B. Rely on source insertion order.

C. ORDER BY band count descending.

D. Assign an explicit numeric rank and sort by it.

10. Which interpretation is justified by a correct band report?

A. The labels prove why a record took longer.

B. The SQL implements the supplied interval rules for the inspected data.

C. The report predicts the next processing duration.

D. The band boundaries are universally optimal.

11. A production job is exactly 120 seconds. Under Brief [0, 120) and Steady [120, 300), which label applies?

A. Brief.

B. Both Brief and Steady.

C. Steady.

D. No band until 121.

12. The explicit Steady interval is changed to [120, 360), but Long still starts at 300. What does a membership-count probe show for 301?

A. Two memberships in the interval definitions.

B. One membership, because CASE can return only one label.

C. A missing-value state.

D. Zero memberships.

13. Quiet jobs are 0, 60 and 119 seconds. A category-driven summary uses LEFT JOIN from all four bands. Which count expression gives zero for the three empty bands?

A. COUNT(classified.job_id).

B. COUNT(*).

C. COUNT(band_order.band).

D. COUNT(band_order.band_rank).

14. A 119.9-second observation is rounded to 120 before classification. What changed?

A. Only formatting; the band necessarily stays Brief.

B. The raw observation became invalid.

C. The rule input changed, moving Brief to Steady.

D. The upper threshold became 119.9.

15. The proposed Steady upper boundary changes from 300 to 360. Raw durations are fixed. Which conclusion follows from fewer Long labels?

A. Processing got faster.

B. Source jobs were necessarily removed.

C. The original measurements were incorrect.

D. The rule reclassified some unchanged observations.

16. There are 12 intake probes: 8 valid, 2 missing and 2 negative. Two valid probes are Review. What is Review as a percentage of valid probes?

A. 16.67%.

B. 20%.

C. 75%.

D. 25%.

17. A valid population is empty. How should a rate with that population as denominator be interpreted?

A. Zero percent, proving no delayed jobs.

B. Undefined, represented as NULL or an explicitly labeled not-applicable display.

C. One hundred percent, because every observed job meets the rule.

D. Use the prior population size without disclosure.

18. Which first-match CASE mutation is caught by J01=0 but not by production row-count reconciliation?

A. Putting elapsed_seconds<900 THEN Long first.

B. Adding ORDER BY job_id.

C. Retaining raw elapsed_seconds in the output.

D. Using a consistent alias for band.

19. H07 has age_hours 72. Fresh [0, 6), Waiting [6, 24), Aged [24, 72), Review [72, infinity) are supplied. What can be concluded?

A. The hold began exactly three calendar days ago.

B. H07 belongs in Review under the supplied age rule.

C. The borrower must be contacted now.

D. H07 is Aged because 72 is its upper endpoint.

20. The Day and Night summaries use the same rule. Which reconciliation checks both grouping directions?

A. Compare only the largest count in each shift.

B. Check shift totals against source shifts and summed band totals against the overall summary.

C. Compare only the number of output columns.

D. Verify both shifts have equal totals.

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 aggregates
https://www.sqlite.org/lang_aggfunc.html

SQLite CREATE TABLE
https://www.sqlite.org/lang_createtable.html

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