Cooling System Failure:

Investigation Through Reconstruction

Seekers Spirits | Cambodia
June 2023

What this document is

This document investigates the cooling system failure that caused Batch 50 to be quarantined in June 2023 through five SQL queries. Each query answers a specific operational question about the run. Each is followed by its output, the findings from that output, and a reliability principle drawn from it.

The investigation depends on reconstructed data. The 2000L still dashboard displayed distillate temperature in real time during production, but none of it was recorded. When Batch 50 failed, there was no retained run data to examine, only a single temperature reading taken after the run had already finished. The still_temperature_logs table reconstructs what happened during the run using a 1–2°C per hour drift later measured directly from the coolant reservoir.

The queries should be read in order. The first establishes the control baseline. Each one after that narrows the investigation: from divergence, to threshold crossing, to the gap between the thermal record and the quality control (QC) result, to what the system knew without the reconstruction.

Table of Contents:

Query 1: Control Baseline

Operational Question

"What did normal behaviour look like across a full run in the 2000L still?"

Batches 48 and 49 are the only completed runs on the 2000L still before batch 50. Both were Orange Liqueur runs completed in May 2023, each lasting 7.5 hours. Both passed quality control. They are the only confirmed examples of full runs on the 2000L still that produced acceptable output, which makes them the baseline for what normal operation looked like.

Batch 50 (the failed batch) ran for 12 hours, longer than either control run, and is where the temperature divergence appears. This query pulls the temperature data from batches 48 and 49 to establish that baseline before comparing it against batch 50. That data, like all temperature data in this document, was reconstructed after the incident.

No temperature limit had been defined for the 2000L still at the time. The 50°C reference used in this analysis comes from the highest temperature recorded on the 300L still during runs that passed quality control. The 300L still was the legacy system with an established operating history. Its temperature range was the only documented reference available for a still operating in the same distillery under the same conditions.

Query:

```
SELECT
    stl.batch_id,
    b.product_name,
    b.batch_date,
    stl.run_hour,
    stl.coolant_temp_c,
    stl.ambient_temp_c,
    stl.notes
FROM still_temperature_logs stl
JOIN batches b ON stl.batch_id = b.batch_id
WHERE stl.batch_id IN (48, 49)
ORDER BY stl.batch_id, stl.run_hour;
```

Output:

batch_id	product_name	batch_date	run_hour	coolant_temp_c	ambient_temp_c	notes
48	Seekers Orange Liqueur	2023-05-08	1	31.00	31.50	Run start.
48	Seekers Orange Liqueur	2023-05-08	2	32.20	31.50	Tasted. Clean.
48	Seekers Orange Liqueur	2023-05-08	3	33.50	31.50	
48	Seekers Orange Liqueur	2023-05-08	4	34.50	31.50	
48	Seekers Orange Liqueur	2023-05-08	5	35.50	31.50	
48	Seekers Orange Liqueur	2023-05-08	6	36.60	31.50	
48	Seekers Orange Liqueur	2023-05-08	7	37.80	31.50	Tasted. No issues.
48	Seekers Orange Liqueur	2023-05-08	8	38.50	31.50	Run complete. QC pass.
49	Seekers Orange Liqueur	2023-05-15	1	30.80	32.00	Run start.
49	Seekers Orange Liqueur	2023-05-15	2	32.00	32.00	Tasted. Clean.
49	Seekers Orange Liqueur	2023-05-15	3	33.40	32.00	
49	Seekers Orange Liqueur	2023-05-15	4	34.60	32.00	Tasted. No issues.
49	Seekers Orange Liqueur	2023-05-15	5	35.80	32.00	
49	Seekers Orange Liqueur	2023-05-15	6	37.00	32.00	
49	Seekers Orange Liqueur	2023-05-15	7	38.20	32.00	
49	Seekers Orange Liqueur	2023-05-15	8	39.10	32.00	Run complete. QC pass.

Key: Blue → Batch 48 (08 May 2023) | Green → Batch 49 (15 May 2023)

Findings

Both control batches show a gradual increase in coolant temperature, starting just above 30°C and peaking in the 38–39°C range. Neither run approaches the 50°C threshold, and both pass quality control. They are the only completed runs on the 2000L still before batch 50 that passed quality control, so they are the only temperature reference for normal operation.

Both control runs ended at 7.5 hours. When batch 50 showed the same pattern, I treated it as normal. The control runs did not capture what happens beyond 7.5 hours. Batch 50 ran for 12 hours.

Reliability Principle

You cannot judge whether a run is abnormal unless the baseline matches the same conditions. Here, the baseline only covered shorter runs. It did not show what happened over longer durations. Because of that, batch 50 looked normal at first.

A baseline is only valid within the conditions it was established in. Applied beyond those conditions, it does not confirm safety. It creates the appearance of it.

Query 2: Divergence Point

Operational Question

"At what hour did Batch 50 first diverge from the control baseline, and how quickly did that divergence increase?"

For each hour of Batch 50, this query compares the observed coolant temperature against the average temperature of the control batches (Batches 48 and 49) at the same point in the run.

The delta_from_baseline column shows the size of that difference. A value near zero means Batch 50 is still close to the control runs. Larger positive values mean Batch 50 is running hotter than the control average by a wider margin.

Query:

```
SELECT stl.batch_id, stl.run_hour, stl.coolant_temp_c,
    ROUND(AVG(ctrl.coolant_temp_c), 2) AS control_avg_temp,
    ROUND(stl.coolant_temp_c - AVG(ctrl.coolant_temp_c), 2) AS
delta_from_baseline
FROM still_temperature_logs stl
JOIN still_temperature_logs ctrl
    ON ctrl.run_hour = stl.run_hour
    AND ctrl.batch_id IN (48, 49)
WHERE stl.batch_id = 50
GROUP BY stl.batch_id, stl.run_hour, stl.coolant_temp_c
ORDER BY stl.run_hour;
```

Output:

batch_id	run_hour	coolant_temp_c	control_avg_temp	delta_from_baseline
50	1	31.00	30.90	0.10
50	2	32.25	32.10	0.15
50	3	33.60	33.45	0.15
50	4	34.8	34.55	0.25
50	5	36.2	35.65	0.55
50	6	37.8	36.80	1.00
50	7	40.0	38.00	2.00
50	8	43.00	38.80	4.20
50	9	46.20	-	-
50	10	49.00	-	-
50	11	51.50	-	-
50	12	54.20	-	-

Coolant Temperature Divergence from Baseline: Batch 50 vs Control

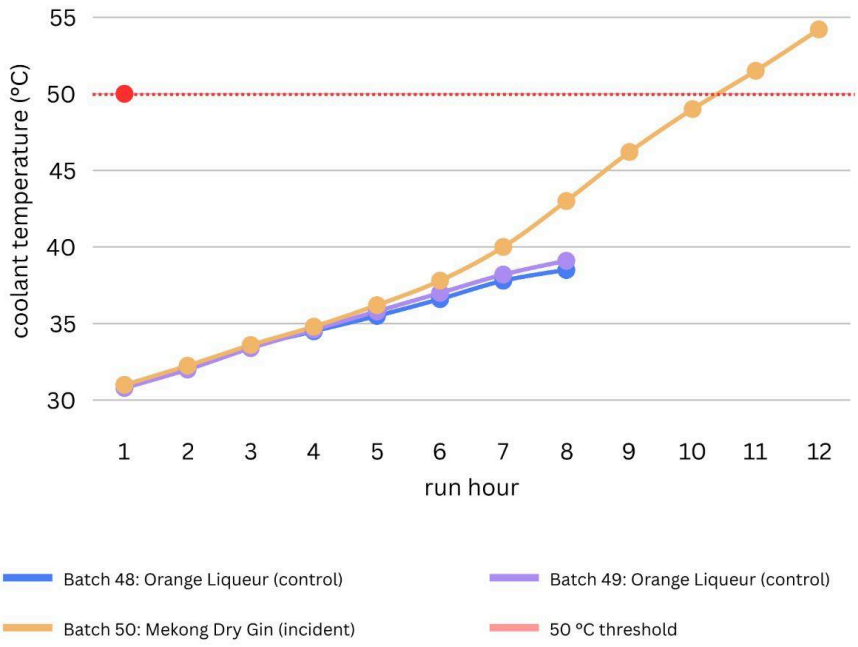


Fig. 1.1: Coolant Temperature Progression - Batch 50 vs Control (Batches 48, 49), showing divergence from the control baseline as the run progressed.

Findings

Batch 50 stays close to the control baseline in the early hours of the run. The comparison ends at hour 8 because the control runs ended at 7.5 hours. After that, there is no baseline for comparison. At hour 1, coolant temperature is 0.10°C above the control average. By hour 4, the gap is 0.25°C and still close to the control range.

The separation starts at hour 5. By hour 6, the delta is 1.00°C. By hour 7, it is 2.00°C. By hour 8, it is 4.20°C. Batch 50 is now moving away from the control runs hour by hour. But with no defined limit and no active monitoring, there was no point at which it would have triggered action.

Reliability Principle

Gradual drift is hard to detect in real time when there is no defined threshold, because each reading can still look acceptable on its own. In practice, this means the system can be moving out of normal conditions before any one reading looks abnormal enough to trigger action. In cases like this, a fixed threshold on its own is not enough to catch the problem early.

Query 3: Threshold Crossing

Operational Question

"At what hour did Batch 50 cross 50°C, and when did the run enter conditions where flavour integrity was compromised?"

This query returns the full hourly temperature profile for Batch 50, with a status label at each hour marking whether conditions were within the normal operating range or above the 50°C threshold.

The threshold crossing marks the point at which the run moved beyond the temperature range associated with acceptable flavour quality.

Query:

```
SELECT stl.run_hour, stl.coolant_temp_c, stl.threshold_breached, stl.notes,
    CASE
        WHEN stl.coolant_temp_c < 50.0 THEN 'Running — no thermal alert'
        WHEN stl.coolant_temp_c >= 50.0 THEN 'Above threshold — flavour
integrity compromised'
    END AS status
FROM still_temperature_logs stl
WHERE stl.batch_id = 50
ORDER BY stl.run_hour;
```

Output:

run_hour	coolant_temp_c	threshold_breached	notes	status
1	31.00	f	Run start.	Running — no thermal alert
2	32.25	f		Running — no thermal alert
3	33.60	f	Tasted. Normal profile.	Running — no thermal alert
4	34.8	f		Running — no thermal alert
5	36.2	f	Tasted. Clean.	Running — no thermal alert
6	37.8	f		Running — no thermal alert
7	40.0	f	Running hotter than usual. Tastes as expected.	Running — no thermal alert
8	43.00	f		Running — no thermal alert
9	46.20	f		Running — no thermal alert
10	49.00	f		Running — no thermal alert
11	51.50	t	Tasted. Clean.	Above threshold — flavour integrity compromised
12	54.20	t	Run complete. Output felt hot — 61.8°C noted from dashboard. Batch held pending QC.	Above threshold — flavour integrity compromised

Findings

Batch 50 crosses the 50°C threshold at hour 11. This is where the run enters the temperature range linked to flavour degradation. Up to that point, the batch stayed below the threshold, even though it had been moving away from the control baseline since hour 5.

Tasting at hour 7 found the profile acceptable, even though the run was already hotter than the control average. Tasting at hour 11 also found no issue, even after the threshold was crossed. In-process tasting did not surface the problem.

Once distillation begins, the batch cannot be corrected. The only point to act is before the still is turned on. After that, the run can be watched, but not improved. Crossing the threshold at hour 11 marks the start of degraded conditions, not a point of recovery.

By hour 12, the run had finished and the output was noted as unusually hot. A temperature of 61.8°C was recorded from the dashboard, and the batch was held pending QC. The flavour issue became clear later during standard QC, after the off notes had developed.

Reliability Principle

In systems where action is only possible upstream, detection downstream comes too late to change the result. By the time the failure is visible, the window to act has already closed.

What matters is whether the problem can be seen early enough to do something about it. If the system only shows the problem after the batch has already been committed, it cannot prevent the loss. It can only show that the batch is already lost.

Query 4: Thermal Record vs Quality Control (QC) Record

Operational Question

"What is the relationship between the temperature log and the formal QC record for Batch 50?"

The 2000L still displayed temperature data during the run, but none of it was saved. When QC took place after the mandatory resting period, the only temperature record available was the informal 61.8°C dashboard note taken after the run.

This query joins the reconstructed temperature log to the QC record. It shows that the run data and the later QC result sat in separate records with no direct link between them. QC could show that batch 50 failed, but not what happened during the run.

Note

QC is carried out weeks after production as standard practice. The problem is that the run conditions were not stored alongside the final quality result.

To show both in one view, the query uses a UNION ALL to combine the 12 hourly temperature entries with the single QC record into one ordered output. The QC record is given sequence 13 so it appears after the final temperature entry. This allows the run to be read in order, from production conditions through to the final assessed outcome.

Query:

```
SELECT * FROM (
    SELECT stl.run_hour AS sequence, stl.coolant_temp_c::text AS
temp_or_outcome,
           stl.threshold_breached::text AS threshold_breached, stl.notes,
'temperature_log' AS source
    FROM still_temperature_logs stl WHERE stl.batch_id = 50
    UNION ALL
    SELECT 13, qc.approved::text, NULL, qc.notes, 'qc_record' AS source
    FROM qc_records qc WHERE qc.batch_id = 50
) combined
ORDER BY sequence;
```

Outcome:

sequence	temp_or_outcome	threshold_breached	notes	source
1	31.00	false	Run start.	temperature_log
2	32.25	false		temperature_log
3	33.60	false	Tasted. Normal profile.	temperature_log
4	34.80	false		temperature_log
5	36.20	false	Tasted. Clean.	temperature_log
6	37.80	false		temperature_log
7	40.00	false	Running hotter than usual. Tasted. As expected.	temperature_log
8	43.00	false		temperature_log
9	46.20	false		temperature_log
10	49.00	false		temperature_log
11	51.50	true	Tasted. Clean.	temperature_log
12	54.20	true	Run complete. Output felt hot — 61.8°C noted from dashboard. Batch held pending QC.	temperature_log
13	false		Muted juniper expression and loss of citrus brightness. Flat botanical profile with a slight cooked character not consistent with house standard. Overall flavour intensity significantly reduced. Batch failed. Cause unknown at time of assessment. Quarantined pending investigation.	qc_record

Findings

Rows 1 to 12 contain temperature data with no QC outcome. Row 13 contains a QC outcome with no temperature data. It was added to allow the production data and QC result to be viewed together. In the database, they are still separate records. There is no direct link between batch 50’s temperature log and its QC result.

The only temperature reference available to QC was 61.8°C, noted informally after the run had already finished. The QC assessment was based entirely on taste, with no record of the run conditions.

The QC notes in row 13 record the cause as unknown at the time of assessment. That is not a gap in the investigation. It shows that the system recorded the failure, but not the conditions that caused it.

Reliability Principle

Data that cannot be connected cannot explain system behaviour. Without a structured way to relate process data to outcomes, the system cannot trace how a given outcome was produced.

When process data and outcome data are not linked, root cause cannot be determined from the record itself. The relationship has to be reconstructed afterwards.

Query 5: The Observability Gap

Operational Question

"Without still_temperature_logs, what could the database tell you about Batch 50?"

This query shows the system as it stood at the time of the incident. Without the reconstructed temperature data, the database contains only one record for Batch 50: a QC entry showing failure.

There is no record of when temperature exceeded 50°C. No hour by hour progression through the run. No indication that the system stayed above threshold for two hours.

What remains is the outcome on its own: a failed batch with notes, but no record of the conditions that produced it.

Query:

```
SELECT b.batch_id, b.product_name, b.batch_date,
       qc.qc_reference, qc.qc_date, qc.approved, qc.notes AS qc_notes
FROM batches b
JOIN qc_records qc ON b.batch_id = qc.batch_id
WHERE b.batch_id = 50;
```

Output

batch_id	product_name	batch_date	qc_reference	qc_date	approved	qc_notes
50	Seekers Mekong Dry Gin	2023-05-29	QC-ISTL-SMDG-230612-1	2023-06-28	f	Muted juniper expression and loss of citrus brightness. Flat botanical profile with a slight cooked character not consistent with house standard. Overall flavour intensity significantly reduced. Batch failed. Cause unknown at time of assessment. Quarantined pending investigation.

Findings

At the time of investigation, the system only showed that batch 50 had failed. Nothing more. There is no temperature progression, no indication of when conditions began to separate, and no record of how long the system operated above the threshold. The failure appears only in the final QC result, after the run had finished.

Two relevant signals existed during the run: distillate temperature was visible on the dashboard, and coolant reservoir temperature was drifting. Only the distillate temperature could be seen at the time, and neither was saved. By the time the investigation began, the only temperature record left was one data point: 61.8°C, taken after the run had finished.

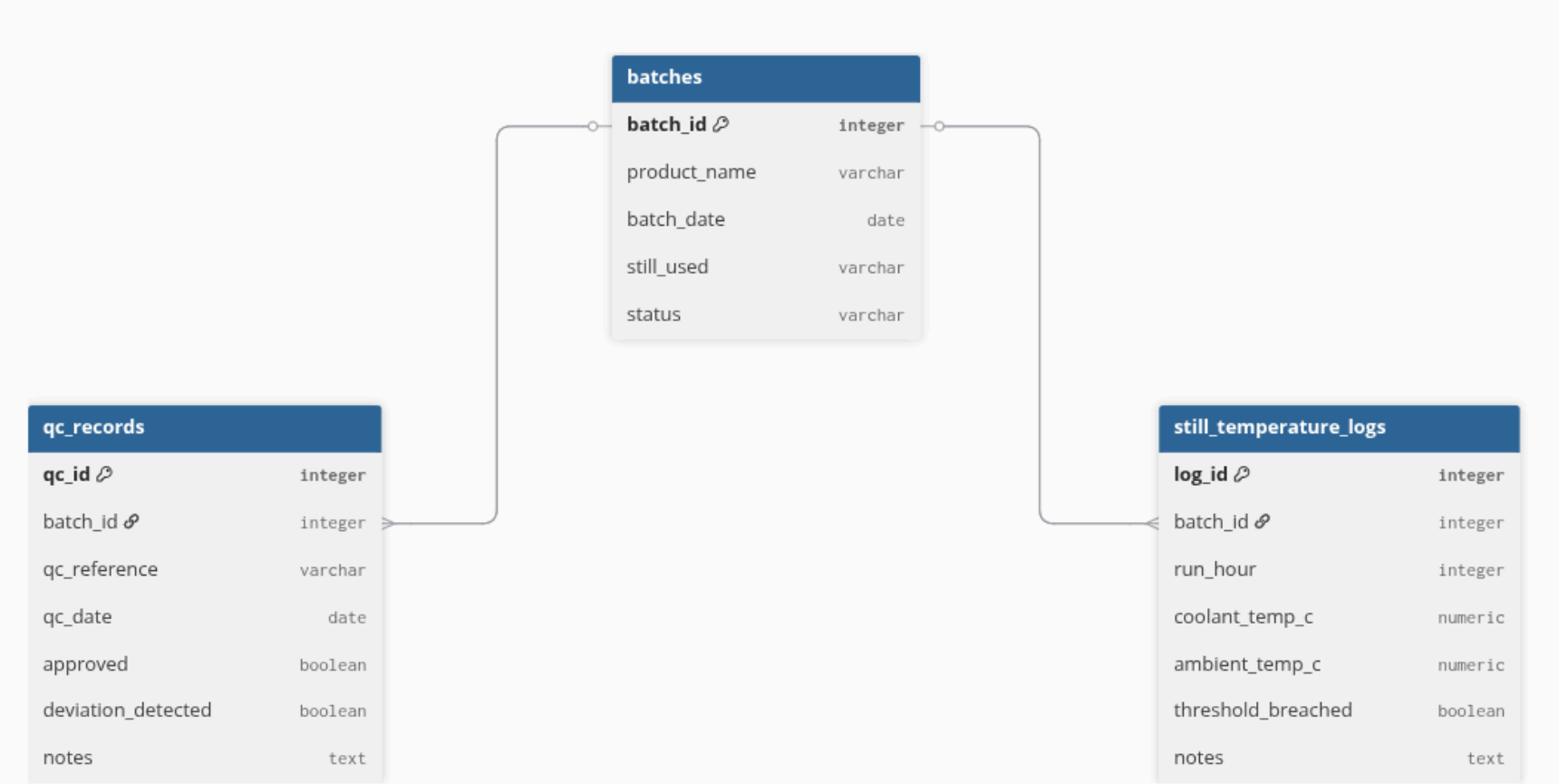
The still_temperature_logs table is a reconstruction. It shows what happened during the run, not what the system actually stored. The queries before this one show what could have been seen if that data had been saved. This query shows the system as it was: it recorded the failure, but not the conditions behind it.

Reliability Principle

Observability requires retention. If data is visible but not recorded, it cannot be used later. A system that captures outcomes without capturing the conditions that produced them cannot explain its own failures. When investigation depends on reconstruction, it is limited by what can be recovered after the fact, and that is rarely complete.

Appendix: Database Schema

Seekers Spirits — Cooling System Investigation: Database Schema



Schema shows only the tables referenced in this investigation.