
◆ **Query Breakdown**

```
_sourceCategory=Labs/Apache/Access (status_code=200 or status_code=404)

| timeslice 1m

| if (status_code = "200", 1, 0) as successes

| if (status_code = "404", 1, 0) as fails

| sum(successes) as success_cnt, sum(fails) as fail_cnt by _timeslice

| (fail_cnt/(success_cnt+fail_cnt)) * 100 as failure_rate_pct

| outlier failure_rate_pct window=5, threshold=3, consecutive=1, direction=+

| where failure_rate_pct_violation > 0
```

---

1. _sourceCategory=Labs/Apache/Access (status_code=200 or status_code=404)

- Pulls Apache Access logs.
- Filters logs where status code is **200 (success)** or **404 (not found / failure)**.
- Keeps the dataset focused on successful vs failed requests.

---

2. | timeslice 1m

- Splits logs into **1-minute intervals**.
- This lets us calculate success/failure ratios over time.

---

3. | if (status_code = "200", 1, 0) as successes

- Creates a **binary column**:
    - If the status code is 200 → assign 1.
    - Otherwise → 0.
- This way, we can later **sum successes** easily.

---

4. | if (status_code = "404", 1, 0) as fails

- Same logic but for failures (404).

- Turns each 404 event into a 1, else 0.

---

5. | sum(successes) as success_cnt, sum(fails) as fail_cnt by _timeslice

- Aggregates per 1-minute slice.
- Counts total **successes vs fails** in each minute.
- Example output:

_timeslice	success_cnt	fail_cnt
10:01	900	15
10:02	920	10
10:03	1000	60

---

6. | (fail_cnt/(success_cnt+fail_cnt)) * 100 as failure_rate_pct

- Calculates **failure rate %**.
- Formula:

\text{failure_rate_pct} = \frac{\text{failures}}{\text{successes + failures}} \times 100

- Example: If 60 fails, 1000 successes → (60/1060)*100 ≈ 5.6% failure rate.

---

7. | outlier failure_rate_pct window=5, threshold=3, consecutive=1, direction=+

This is the anomaly detection step.

- **failure_rate_pct** → metric being monitored.
- **window=5** → looks at the last **5 minutes** as a baseline.
- **threshold=3** → flags if current failure rate is **≥ 3 standard deviations above the mean** of last 5.
- **consecutive=1** → only **1 anomaly point** needed to flag.
- **direction=+** → only detects **spikes upward** (failure rate unusually high).

---

8. | where failure_rate_pct_violation > 0

- Filters results to show **only violations** (when anomaly is detected).

- If the outlier didn't trigger, that row is filtered out.

🔹 Example Log Scenarios Where Outlier is Useful

✅ Scenario 1: Web Server Error Spikes

- Normally 404 failure rate is 1–2%.
- Suddenly jumps to 30% because:
  - A new app deployment broke links.
  - Attackers are probing for missing pages.
- Outlier flags this **spike in failure rate**.

✅ Scenario 2: Authentication Failures

- Use on login logs (status=FAILED).
- Normally 10 failed logins/hour.
- Suddenly 200 failed logins in 5 minutes → possible **brute force attack**.
- Outlier flags the anomaly.

✅ Scenario 3: Database Query Errors

- Monitor DB logs for query errors (error_code).
- If error rate suddenly surges, it may mean:
  - DB misconfiguration.
  - Malicious queries attempting SQL injection.

✅ Scenario 4: Network Device Failures

- Monitor firewall/router logs.
- If packet drops suddenly spike above baseline, Outlier will detect possible **DDoS attack** or hardware issue.

✅ **In short:**

This query calculates **failure rate % of web requests**, then uses **statistical anomaly detection** to automatically flag when the failure rate suddenly spikes above normal baseline behavior.