

Answer Key — DLP While Using AI

Admin / self-check reference

Task 1 — Python stack trace

Class: with-redaction. The trace is useful to share with an AI, but it contains an internal hostname, a real user email, and a bearer token — all of which have to come out before it leaves.

Redacted version:

```
Traceback (most recent call last):
  File "<app>/orders/api.py", line 142, in checkout
    user = db.get_user("<user@example.com>")
  File "<app>/orders/db.py", line 58, in get_user
    conn = psycopg2.connect(DATABASE_URL)
psycopg2.OperationalError: could not connect to server:
  Connection timed out
  Is the server running on host "<db-host>"
  and accepting TCP/IP connections on port 5432?

Request headers:
  Authorization: Bearer <REDACTED_JWT>
  X-Request-Id: <request-id>
```

What was removed:

- `db-prod-3.internal.acme.corp` — internal hostname, reveals environment naming and infra topology. Replaced with `<db-host>`.
- `jane.smith@acme.corp` — real user email, PII and identifies an internal domain. Replaced with `<user@example.com>`.
- `Bearer eyJhbGciOiJIUzI1NiJ9...` — live JWT. Replaced with `<REDACTED_JWT>`, *and the token must be rotated regardless*, because you cannot prove it was not already in flight.
- `/srv/acme/...` — absolute path leaking the company name and deployment layout. Replaced with `<app>`.

- Request ID — low-risk but also low-value for debugging connection timeouts, and worth normalizing for hygiene.

Common mistakes:

- Leaving the domain in the email ("I only removed the name part") — the domain is the leak.
- Leaving the JWT "because it is expired" — verify, do not assume, and rotate regardless.
- Leaving the internal hostname because it "looks harmless" — hostnames reveal environment tiering, naming conventions, and sometimes service discovery structure.
- Leaving the absolute file path — it names the company and the deployment directory in one line.

Task 2 — SQL query

Class: with-redaction, or enterprise-only depending on content. A query that references a real production table and a real customer ID is borderline; in a disciplined organization this belongs inside the enterprise tier at minimum, and it still needs the identifying values redacted before you send it.

Redacted version:

```
SELECT o.order_id, o.total, o.status, o.created_at
FROM orders_table o
WHERE o.customer_id = '<customer_id>'
      AND o.created_at >= '2026-01-15'
      AND o.created_at <  '2026-02-15'
      AND o.status IN ('paid', 'shipped', 'refunded')
ORDER BY o.created_at DESC
LIMIT 100;
```

What was removed:

- `orders_prod_2026` — real production table name, reveals internal naming conventions and partitioning scheme. Replaced with `orders_table`.
- `cust_84312` — real customer identifier tied to a real account. Replaced with `<customer_id>`.

- Date range kept as-is — dates alone are low-risk and the AI needs them to reason about selectivity and index choice.
- Status values kept as-is — they are generic enum values, not identifying data.

Common mistakes:

- Leaving the real customer ID because "it is just a number" — any stable identifier pointing to a real person is PII.
- Leaving the real table name because "it is just a name" — production table names leak schema evolution, partitioning, and sometimes business strategy.
- Redacting the date range unnecessarily, which then prevents the AI from giving useful index advice.
- Aliasing the table to `o` and thinking that hides it — the original name is still on the `FROM` line.

Task 3 — Auth bug report

Class: never. The report contains a live session cookie belonging to a real, named user. A live session cookie is a credential. It does not matter how long it is, how many characters you believe are needed, how expired you think it is, or how "illustrative" it feels — it belongs in the never class and it does not get pasted into an AI, even an enterprise-tier one.

Redacted version (the right fix is to describe the bug, not quote it):

A user reports that `POST /sessions/refresh` returns HTTP 401 even though the session cookie appears to still be well inside its configured TTL. A fresh login works, but the refresh endpoint consistently fails for this user. The behavior is reproducible.

What are common causes of a session-refresh endpoint rejecting a cookie that is still within its TTL? I am especially interested in clock skew, rotating-secret issues, and server-side session-store invalidation.

What was removed:

- `alice@acme.corp` — real user email, identifies a person and an internal domain. Dropped entirely; the bug description does not need a name.

- `SESSION=abc123xyz...` — the live session cookie. Dropped entirely; a cookie is a credential, not illustration material.
- `/api/v2/sessions/refresh` — narrowed to `/sessions/refresh`. The model does not need the full internal path structure to answer the question.
- `@platform-core` and `@identity-team` — internal Slack handles reveal org structure. Dropped entirely.
- "exported cookie jar" — removed, because it implies a workflow that itself is risky.

Common mistakes:

- Pasting the cookie "to illustrate the bug shape" — do not. Describe the shape in words. An AI does not need the real value to reason about the failure mode.
- Including the Slack handles because "they are just names" — they map internal team structure, which is enterprise-tier data at best, and they add nothing to the question.
- Redacting only the cookie value but leaving the user email — you still identified the user and the domain.
- Putting the redacted version into a personal AI account because "the real secrets are gone" — the four-class model is about the original content, not the redacted copy; once something was in the "never" class, the safer move is to handle the debugging inside an enterprise-tier tool.