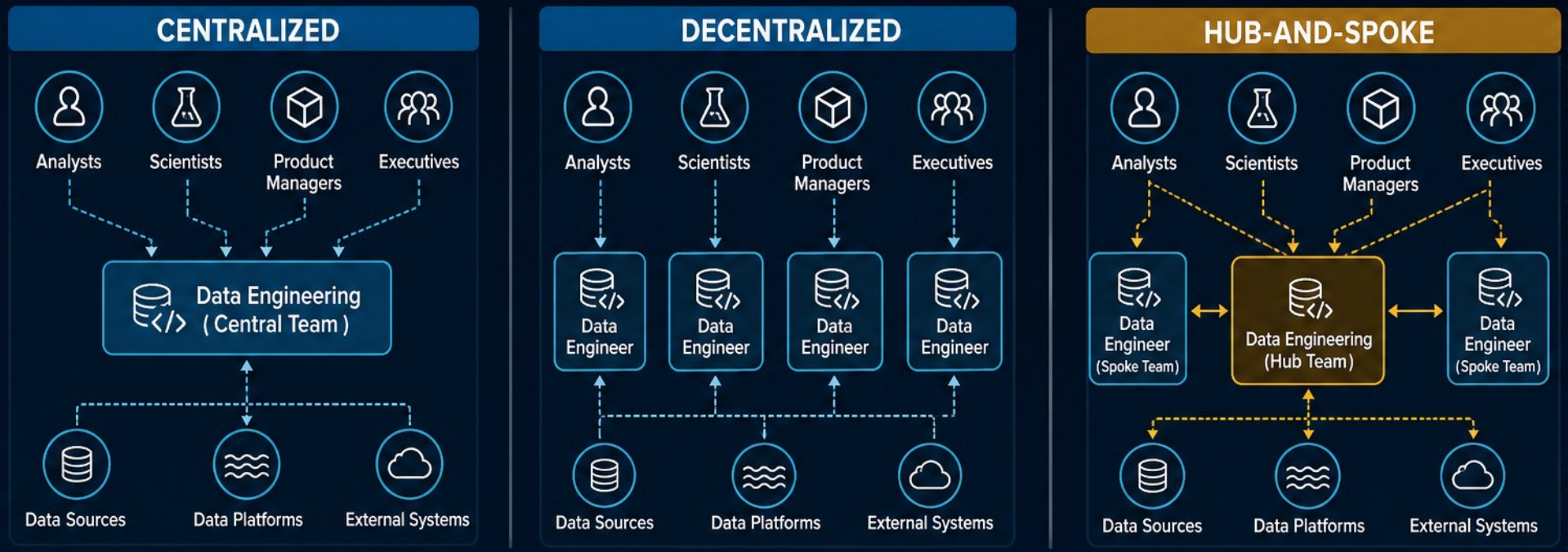
Data Engineering Skills: Core Skills and Practice

- Build a working mental model, not a tool checklist
- 18,000+ US postings: SQL, Python, pipelines are table stakes
- AI raises the judgment bar for data readiness
- Practice: ingest, model, transform, operate, communicate
- End-to-end mini-project: ties SQL, Python, modeling, reliability

What Data Engineering Actually Is

- Design, build, and operate systems turning raw events into trustworthy tables
- Pipelines, warehouse models, platform maintenance, pipeline operations
- Upstream of every dashboard, model, and executive number
- Not data science or analytics: engineers build the road; analysts drive

Where Data Engineers Sit in the Organization



- Three common structures: centralized, decentralized, and hub-and-spoke
- Recommended hiring order: engineer, analytics engineer, analysts, then specialists
- Serve analysts, scientists, product managers, and executives
- Career ladder: L3 junior to L7 principal, with IC or management fork after senior

The Modern Data Stack in One Picture

- Five core layers: ingestion, storage, transformation, orchestration, consumption
- Snowflake, BigQuery, Databricks: choose warehouse or lakehouse by workload
- Batch suits scheduled loads; streaming fits real-time needs
- dbt for transformation, Airflow/Dagster for orchestration
- Avoid over-tooling: start lean, add layers as pain points emerge

SQL: The Highest-Weighted Skill

- SQL appears in ~41% of interview rounds and nearly every working day

- Core fluency: joins, aggregation, window functions, CTEs, NULL handling

- Write maintainable models with clear grain and naming

- Optimize with partitioning, pruning, and execution-plan reasoning

```sql
WITH recent_orders AS (
  -- CTE: isolate recent orders
  SELECT
    o.order_id,
    o.user_id,
    o.order_ts,
    o.amount
  FROM orders o
  WHERE o.order_ts >= CURRENT_DATE - INTERVAL '30' DAY
)
SELECT
  u.user_id,
  u.country,
  ro.order_id,
  ro.order_ts,
  ro.amount,
  SUM(ro.amount) OVER (
    PARTITION BY u.user_id
    ORDER BY ro.order_ts
    ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
  ) AS running_amount,
  LAG(ro.amount) OVER (
    PARTITION BY u.user_id
    ORDER BY ro.order_ts
  ) AS prev_amount
FROM recent_orders ro
JOIN users u
  ON ro.user_id = u.user_id
WHERE u.country IS NOT NULL;
```

CTE
Structure and clarity

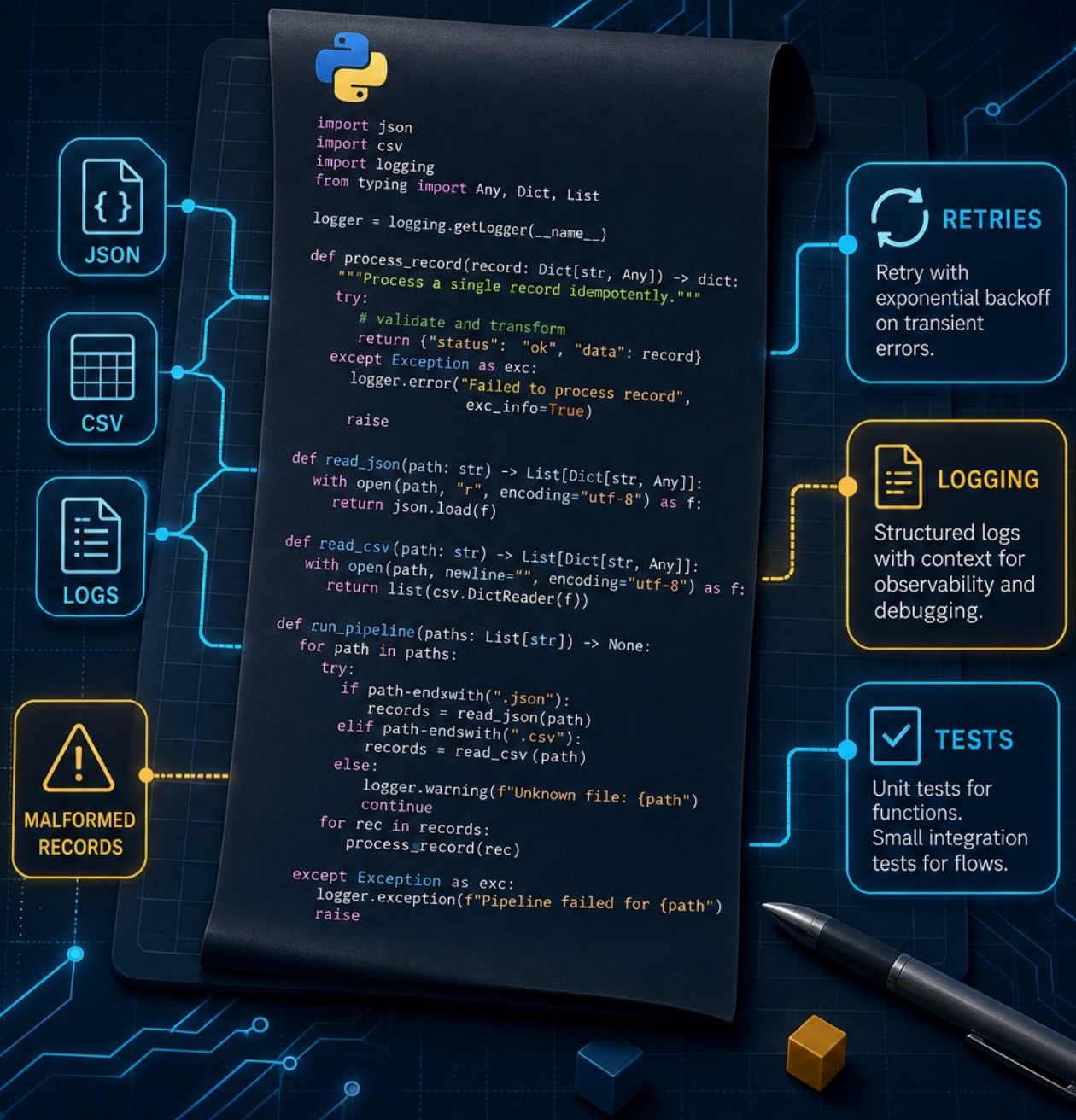
JOIN
Combine related data

WINDOW
Running totals and lookbacks

NULL
Handle missing values

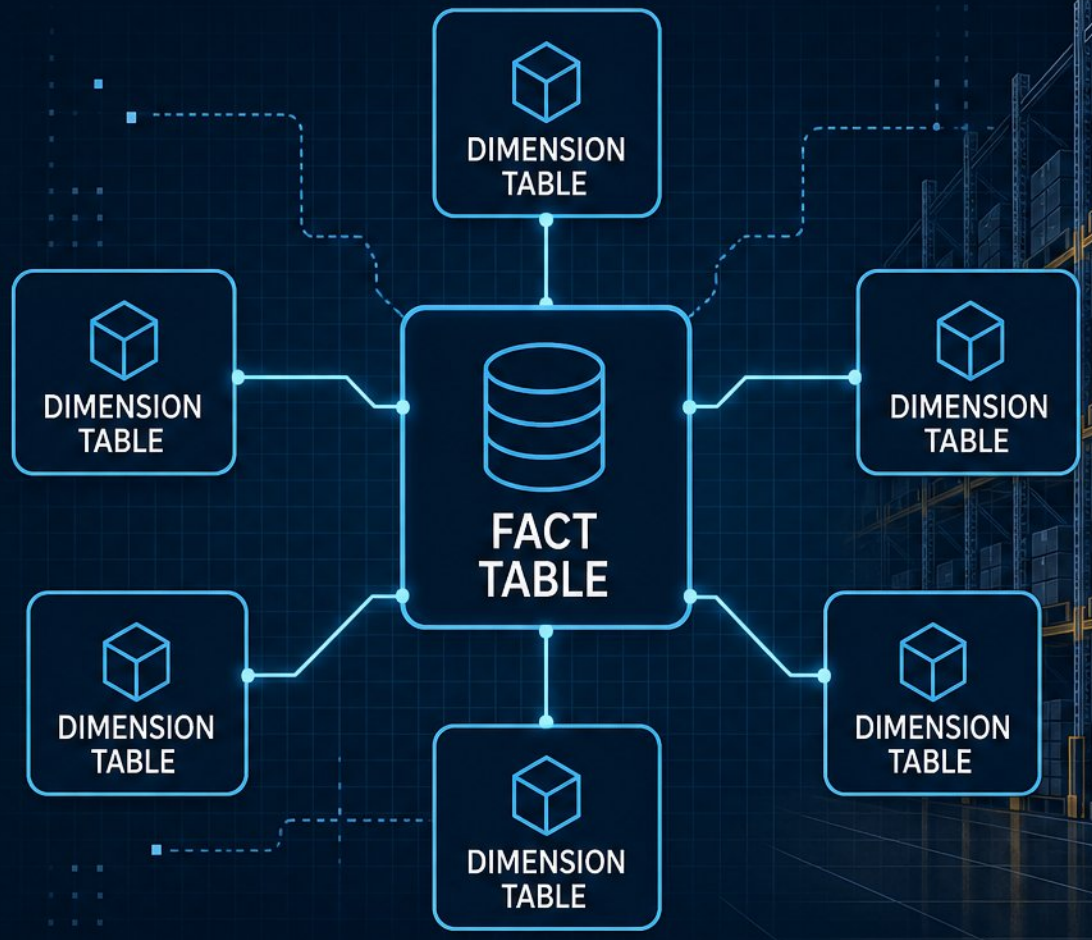
Python: Pipeline Glue, Not LeetCode

- Appears in ~71% of job postings, but interviews test data manipulation, not algorithms
- Core work: reading JSON, CSV, logs, and malformed records
- Production habits: error handling, logging, retries, idempotent functions
- Test pipeline code with unit tests and small integration tests
- AI raises the bar: judgment on correctness and idempotency matters more



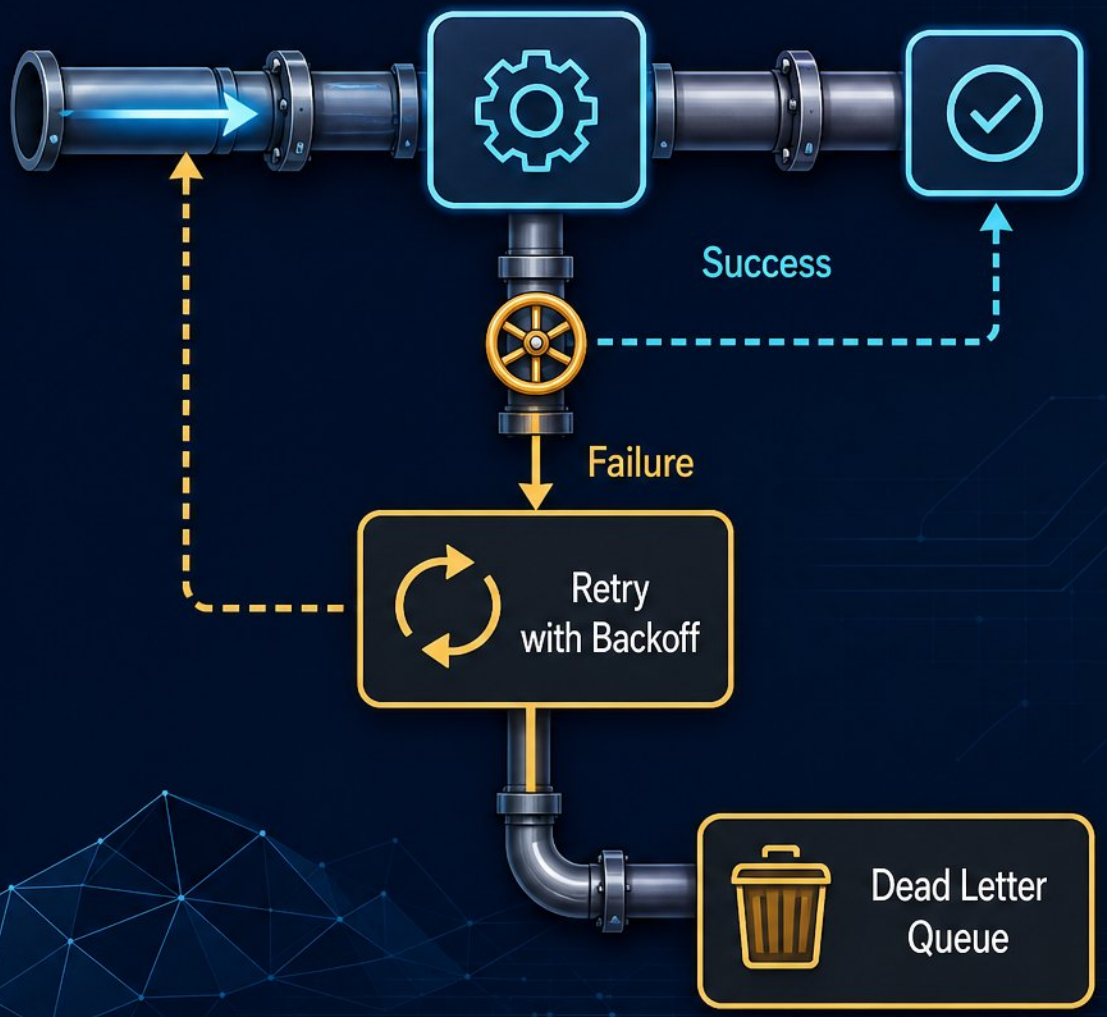
Data Modeling and Schema Design

- - Dimensional modeling: facts, dimensions, and defining grain
- - Star vs. snowflake vs. wide tables: when to denormalize
- - Slowly changing dimensions: Type 1, Type 2, and hybrids
- - Schema evolution strategies and common anti-patterns
- - Modeling judgment separates mid-level from senior engineers



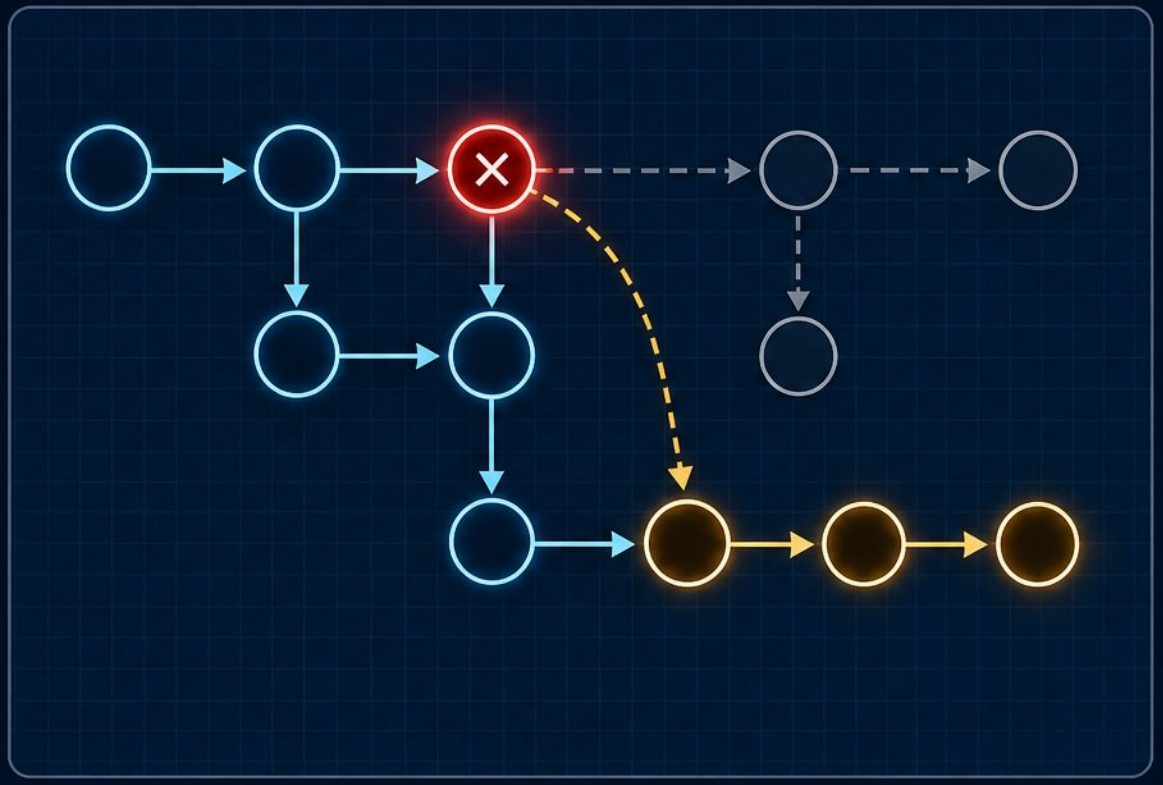
Building Reliable Pipelines

- Idempotency: safe reruns, retries, and backfills without duplicates
- Parameterized dates, not hardcoded NOW(): backfill any period
- Dead letter queues + retries with backoff keep pipelines running
- At-least-once with deduplication beats expensive exactly-once



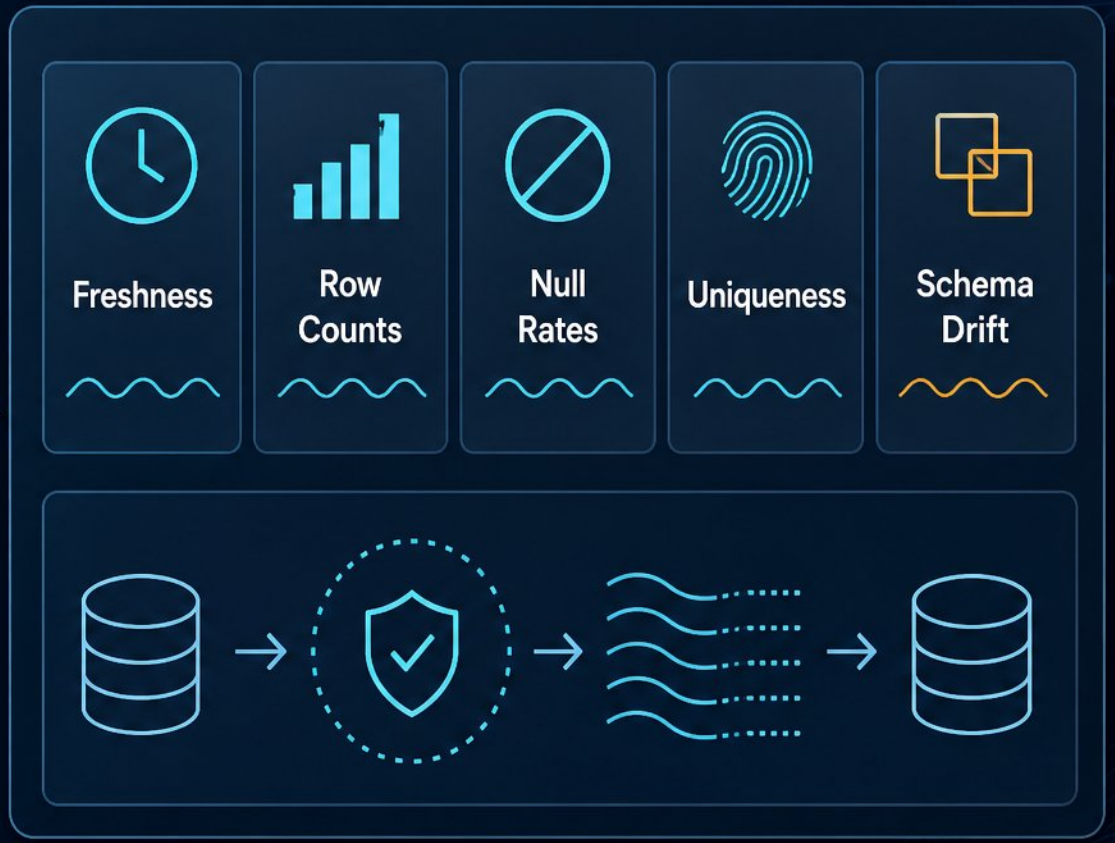
Orchestration and Failure Recovery

- DAGs model task dependencies and execution order
- Cron jobs stop scaling after a few pipelines
- Parameterized dates enable safe backfills and catchup
- Backfills handle bug fixes, schema changes, and late-arriving data



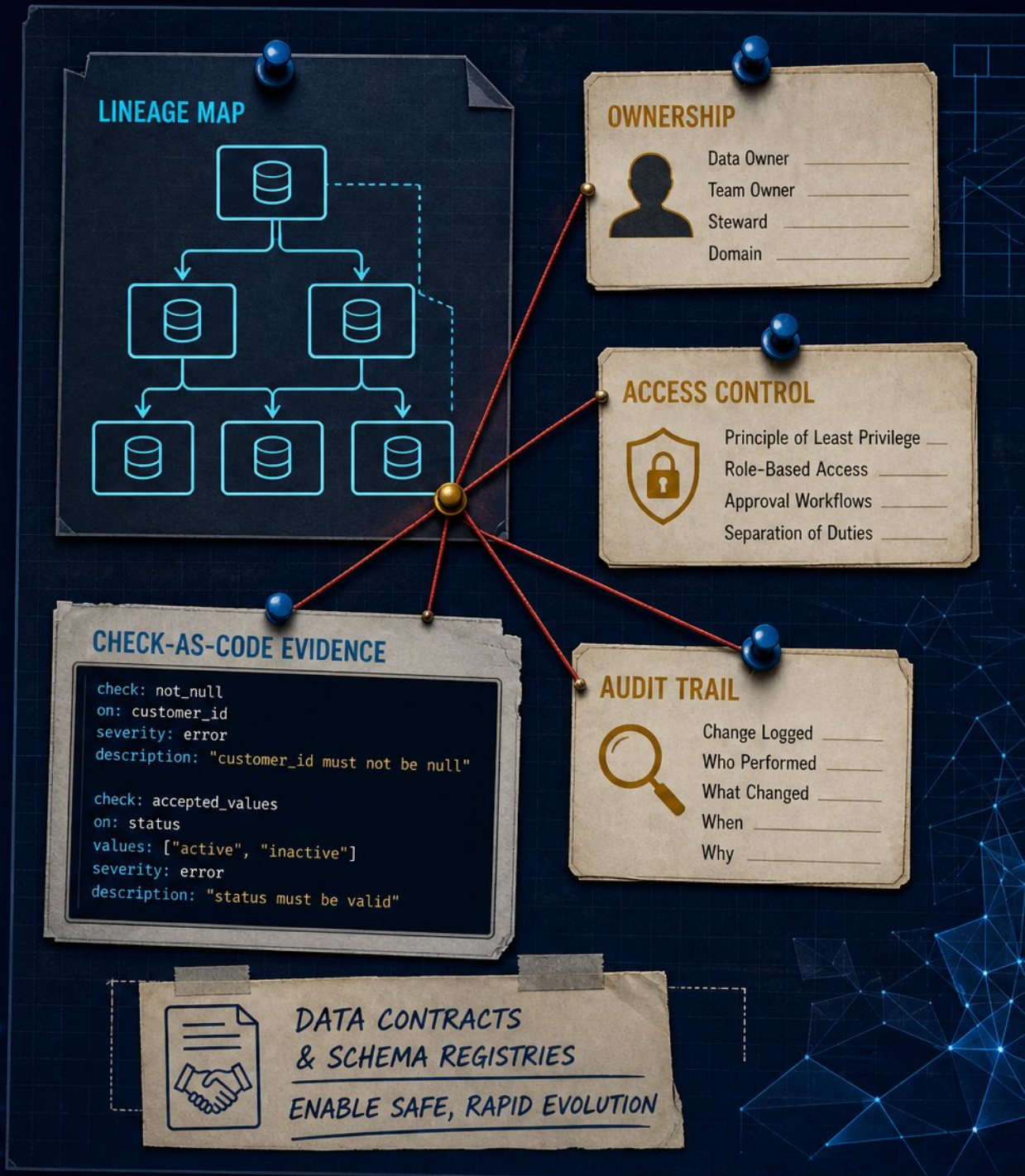
Monitoring, Data Quality, and Observability

- - Track execution health and data quality, not just job success
- - Key signals: freshness, row counts, null rates, uniqueness, schema drift
- - Validate at the boundary: catch bad data at ingestion
- - Alert thresholds based on business impact
- - Track MTTD and MTTR to improve response times



Governance Without Slowing Delivery

- Lineage, documentation, ownership, and access control as production practices
- Check-as-code, versioned models, and audit trails for practical governance
- Data contracts and schema registries enable safe, rapid evolution
- AI-era: richer metadata and stronger controls for agentic systems



Collaboration and Production Practices

- Version control all pipeline code, models, and configuration
- Enforce code review, shared standards, and CI/CD for pipelines
- Communicate clearly with analysts, scientists, and platform teams
- A build-and-operate mindset beats certifications: evidence of shipped, long-running systems is what counts

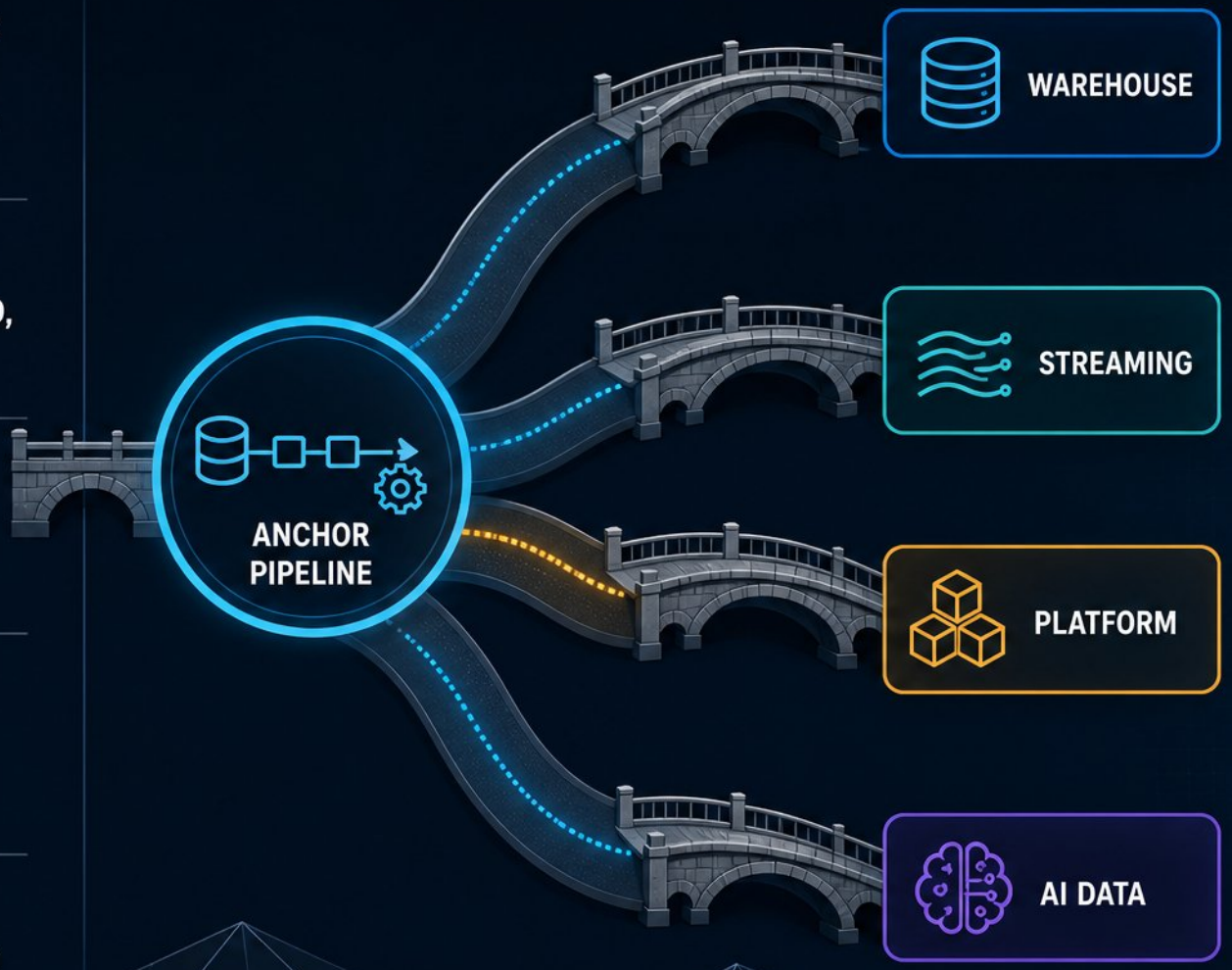
Putting Core Skills into Practice

ONE ANCHOR PROJECT: END-TO-END DATA PIPELINE

- Build one end-to-end anchor project: ingest, store, transform, test, orchestrate, monitor, serve
- Prove implementation with idempotent reruns, quality checks, and runbooks
- Apply SQL, Python, and modeling together in a realistic scenario
- Evaluate trade-offs: batch vs. streaming, modeling choices, and reliability decisions

Next Steps for Practice and Growth

- Build one anchor pipeline: ingest, store, transform, test, orchestrate, monitor, serve
- Add 2–3 boosters: data quality checks, CI/CD, CDC, or streaming
- Package work: diagram, README, run command, tests, runbook
- Choose specialization: warehouse, lakehouse, streaming, platform, or AI data
- Practice: time-box SQL, Python, modeling, design problems; explain decisions aloud



PersonWise.

PersonWise • AI digital-human course platform

Scan the code or click anywhere to watch this course live, with voice Q&A

personwise.ai/t/1038cbc5/public