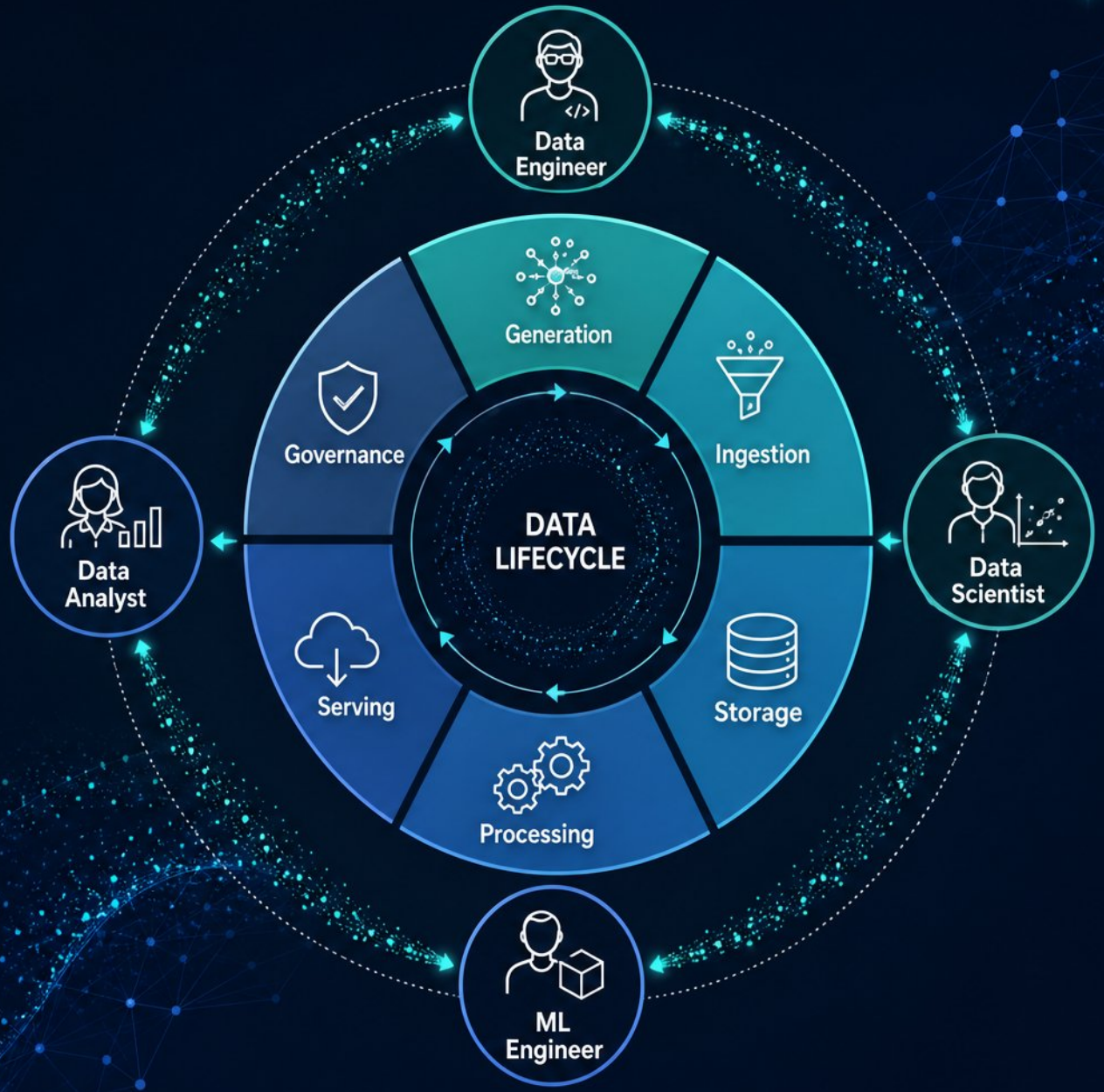
Data Engineering Fundamentals

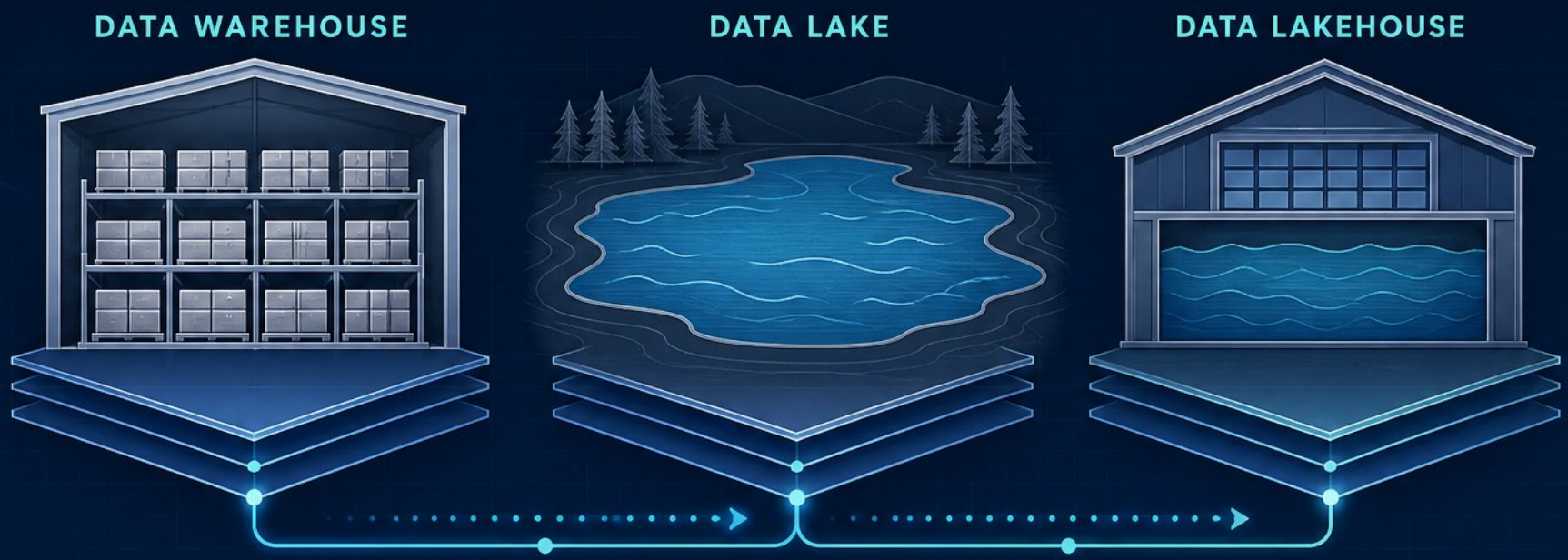
- Design, build, and operate systems for reliable data delivery
- Supports analytics, ML, and decision-making with correct, timely data
- Course covers landscape, architecture, storage, SQL, modeling, and quality
- Includes a hands-on end-to-end pipeline project
- Outcomes: lifecycle understanding, architecture comparison, and a working pipeline

The Data Engineering Landscape and Core Roles

- Data engineer builds reliable data supply; analyst interprets; scientist predicts; ML engineer serves models
- Lifecycle: generation, ingestion, storage, processing, serving, governance
- Smaller firms favor generalists; larger firms use specialized pipeline, warehouse, platform, and streaming roles
- Accountable for pipeline SLAs, correctness, freshness, cost, and quality
- AI-native tooling and data-as-a-product shift focus toward reliability, governance, and meaningful transformations

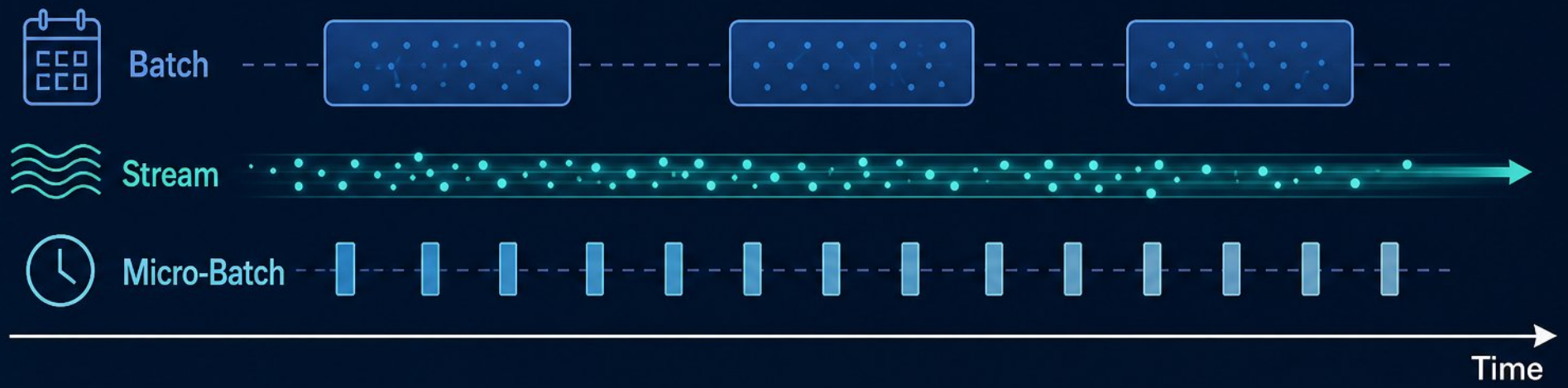


Warehouses, Lakes, and Lakehouses



- **Data warehouse:** structured, schema-on-write, fast governed BI and SQL analytics
- **Data lake:** raw data on low-cost storage, schema-on-read, flexible for ML
- **Lakehouse:** lake economics with ACID transactions, schema enforcement, and performance
- **Warehouses** lack ML flexibility; lakes lack governance and ACID guarantees
- Choose a lakehouse for unified BI and ML, or combine patterns

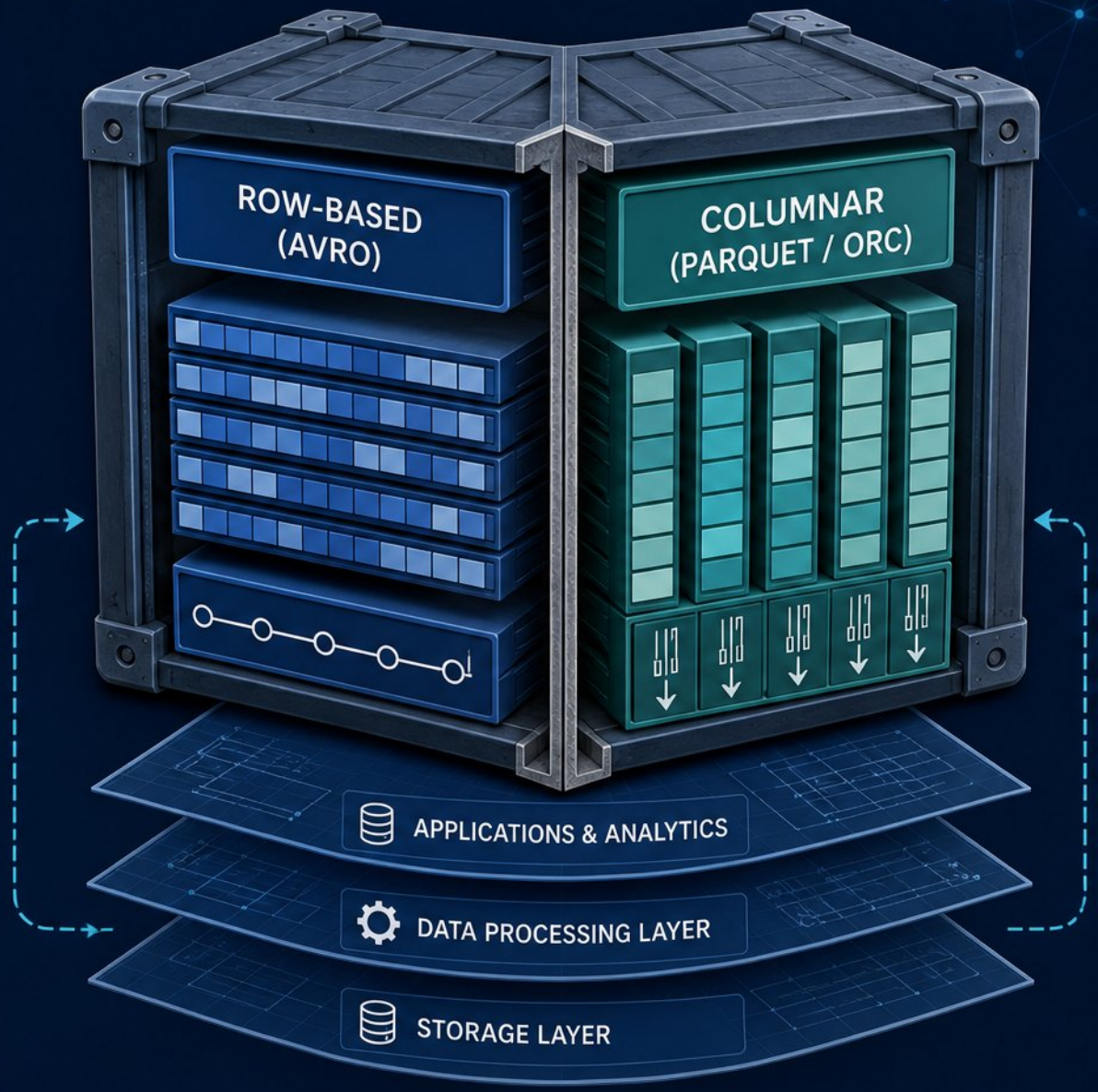
Batch, Stream, and Micro-Batch Processing



- Batch: processes bounded datasets on a schedule for reports, ML training, and backfills
- Stream: handles unbounded events as they arrive for fraud, IoT, and live dashboards
- Micro-batch: processes small groups every few seconds for near-real-time use cases
- Stream adds complexity in state, cost, and failure recovery; batch is simpler and cheaper
- Decision rule: start from the business deadline, not the tool

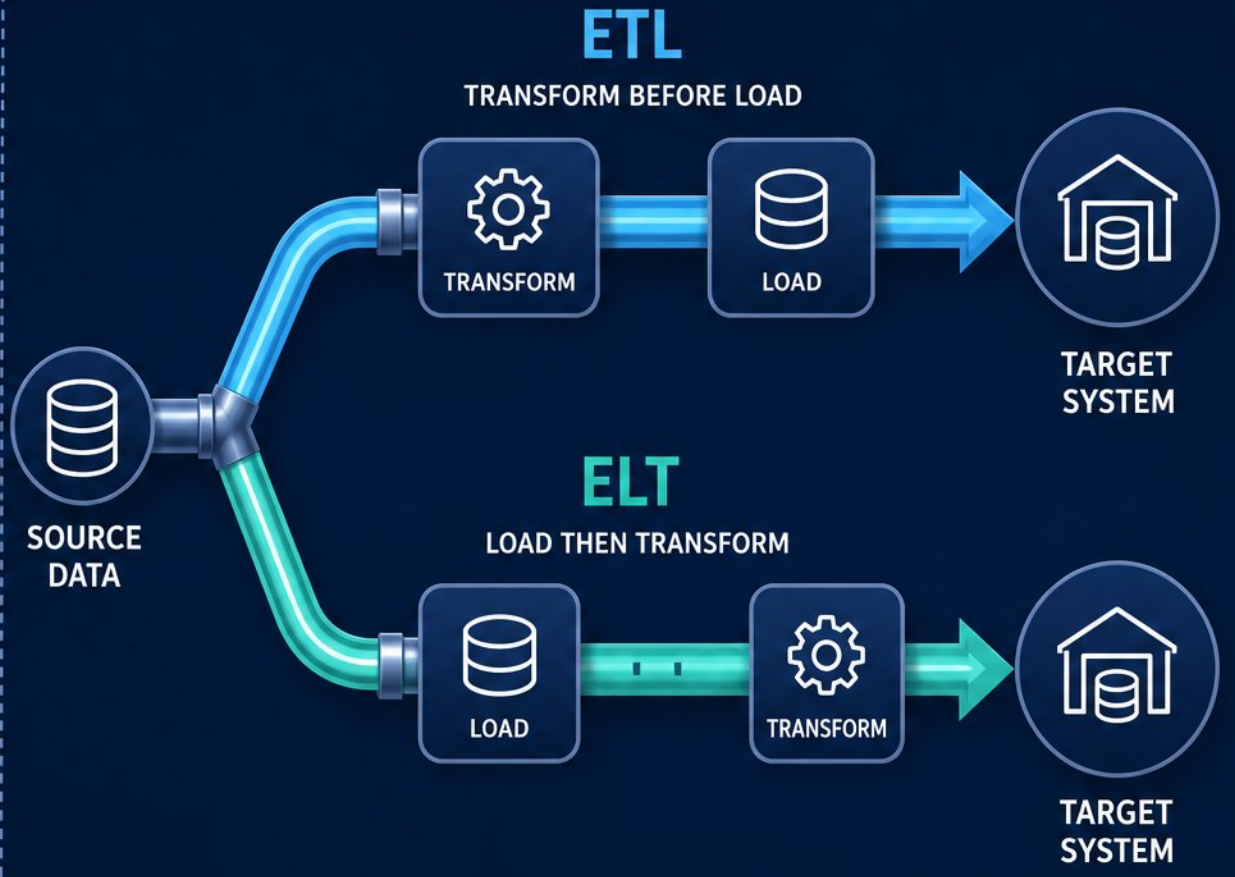
Core Data Storage Concepts and File Formats

- Relational for transactional integrity; NoSQL for flexible high-volume workloads
- Columnar stores enable fast scans and pruning for analytics
- Parquet and ORC: columnar formats optimized for reads
- Avro: row-based format for streaming and schema evolution
- Recommended pattern: Avro for events, Parquet for analytical storage



ETL and ELT Fundamentals

- ETL transforms before loading; ELT loads raw, transforms later
- ELT is standard for cloud warehouses and lakehouses today
- Choose ETL for privacy, compliance, or target-system constraints
- Common transformations: cleaning, casting, joining, filtering, aggregating
- Orchestrators like Airflow, Dagster, Prefect manage pipelines



SQL for Data Engineering

- SQL: primary interface for query, transform, and validate data
- Essential skills: joins, CTEs, window functions, date logic
- Engineering SQL: reproducible, tested, incremental, modular
- Cleaning tasks: dedupe, null handling, casts, integrity checks
- Interview reality: most-tested data engineering skill

```sql
WITH recent_orders AS (
  SELECT
    o.order_id,
    o.customer_id,
    o.order_date,
    o.amount
  FROM orders o
  WHERE o.order_date >= CURRENT_DATE - INTERVAL '30 day'
)
SELECT
  c.customer_id,
  c.customer_name,
  r.order_id,
  r.order_date,
  r.amount,
  ROW_NUMBER() OVER (
    PARTITION BY c.customer_id
    ORDER BY r.order_date DESC
  ) AS order_rank
FROM recent_orders r
JOIN customers c
  ON r.customer_id = c.customer_id
WHERE c.is_active = TRUE
ORDER BY c.customer_id, r.order_date DESC;
```

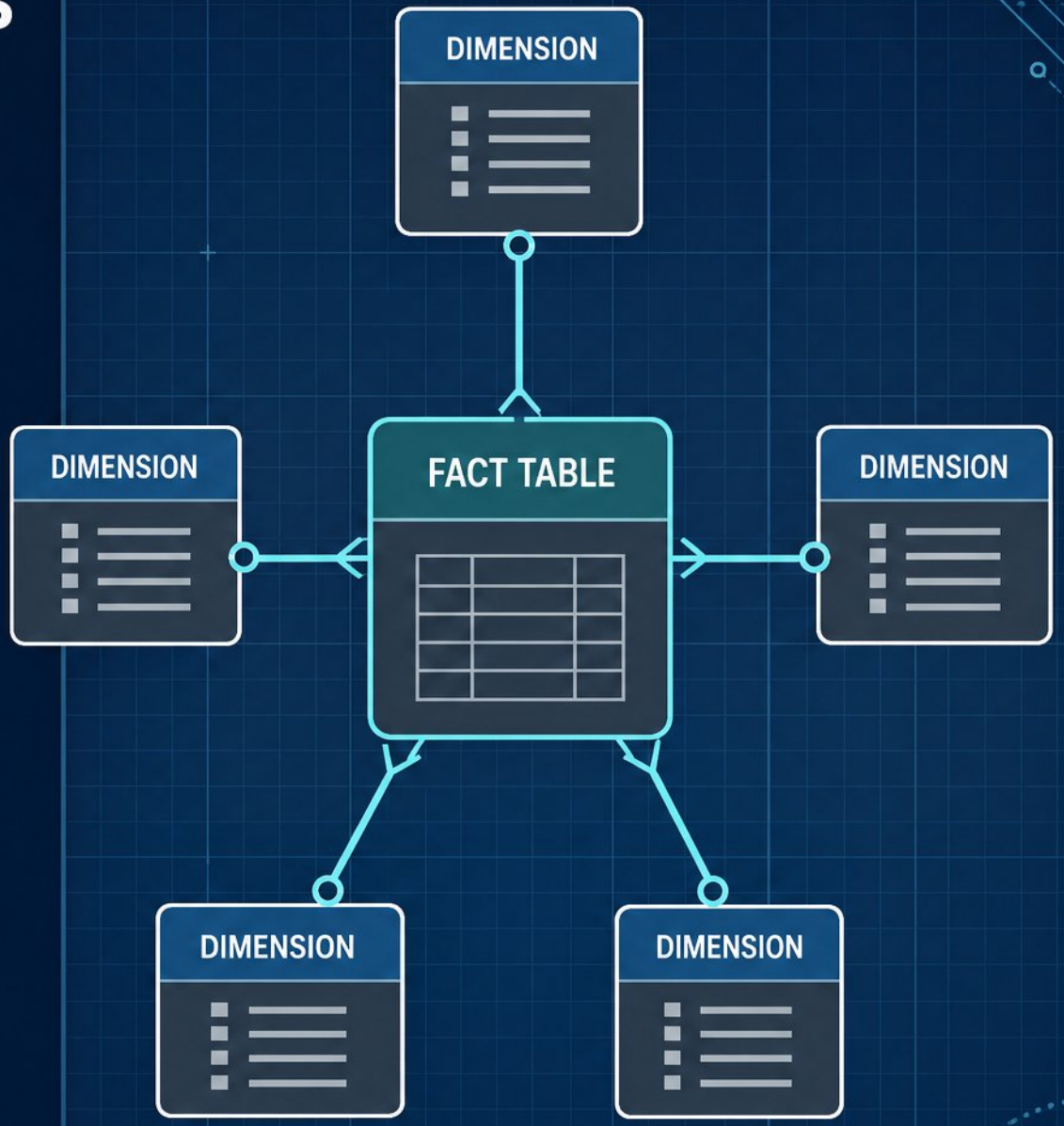
CTEs
Break complex logic into readable steps

Joins
Combine data from multiple sources

Window Functions
Analyze rows across a partition

Data Modeling Basics

- Facts = numbers you measure; dimensions = context you filter by
- Star schema: central fact table linked to denormalized dimensions
- Grain defines what one fact row represents
- Star schemas trade redundancy for simpler, faster analytical queries
- Surrogate keys and slowly changing dimensions support accurate history



Data Quality Essentials

- Completeness, uniqueness, consistency, timeliness, validity, and accuracy
- Validity checks expected formats; completeness checks required fields are present
- Pre-publication checks: nulls, duplicates, ranges, freshness, referential integrity
- Quality monitoring checks values; observability checks pipeline health and schema drift
- Match thresholds to primary consumers and business purpose

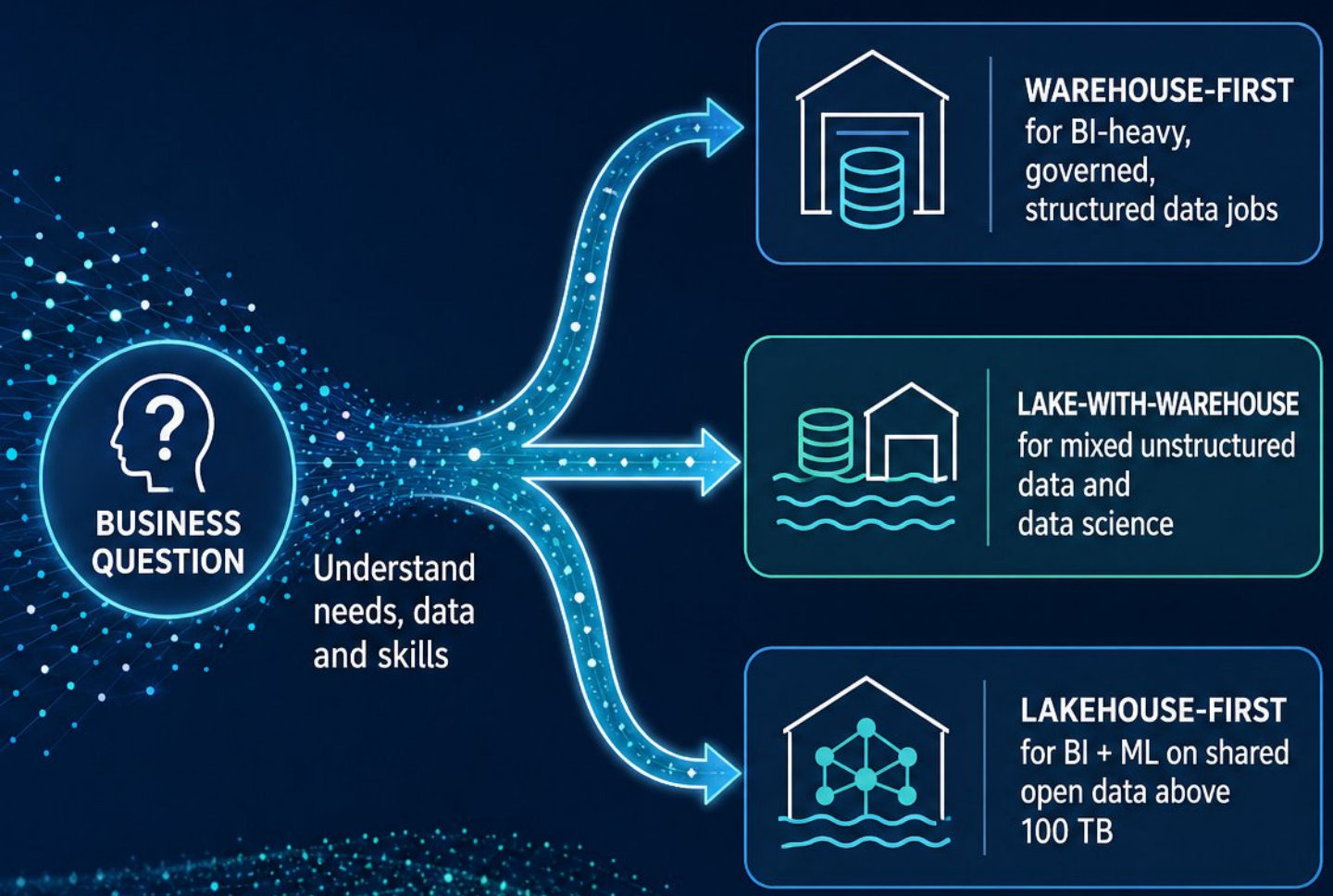
Governance, Metadata, and Access Control

- Governance secures, documents, and makes data trustworthy across the lifecycle
- Metadata describes structure, meaning, origin, and usage; lineage tracks pipeline changes
- Access control enforces permissions at platform, dataset, column, and row levels
- Data catalogs help consumers discover, understand, and trust available datasets
- Start lightweight: define owners, document critical datasets, apply basic access rules

Modern Data Engineering Tooling

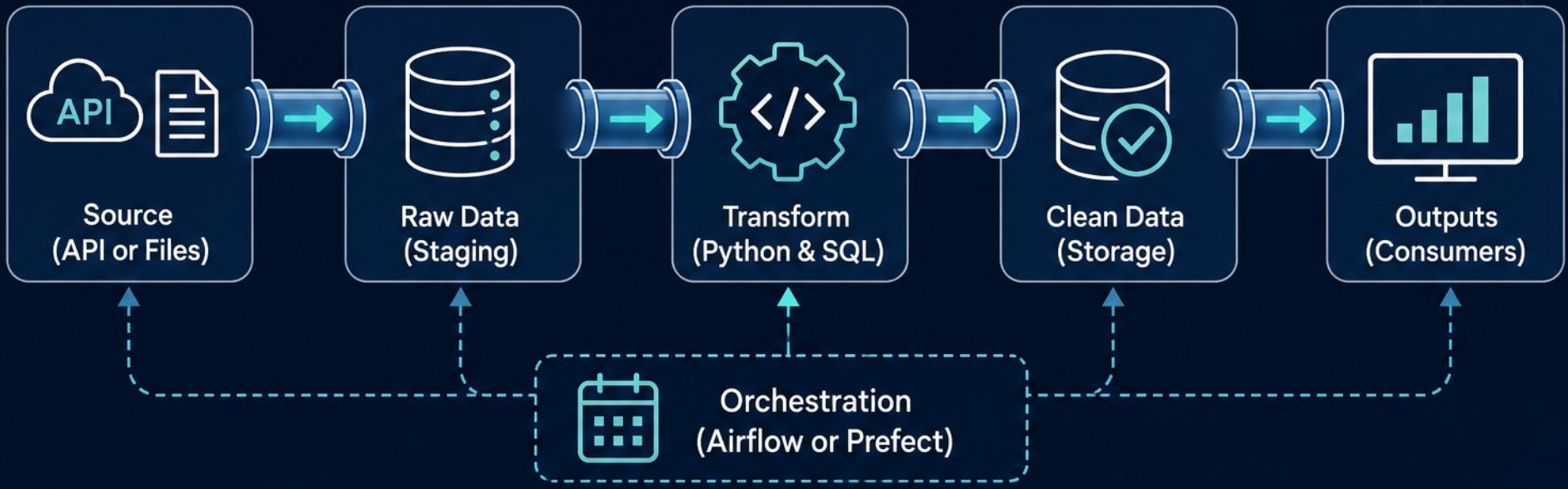
- Modular stack layers: ingestion, storage, transformation, orchestration, quality, and serving
- Ingestion: Fivetran and Airbyte for connectors, Kafka for event streams, dlt for Python loading
- Storage: Snowflake, BigQuery, Databricks, and open formats like Iceberg and Delta Lake
- Transformation with dbt; orchestration with Airflow, Dagster, or Prefect
- Evaluate by category, cloud fit, operational cost, pricing, and team skills

Choosing the Right Architecture and Tools



- Match architecture to business use, data variety, latency, and team skills
- Warehouse-first for BI-heavy, governed, structured data jobs
- Lake-with-warehouse for mixed unstructured data and data science
- Lakehouse-first for BI + ML on shared open data above 100 TB
- Think decision-first: questions, sources, skills, not a universal solution

Building Your First Data Pipeline



- - Extract from an API or files, load raw data into staging
- - Clean and transform with Python and SQL, then store results
- - Keep raw data immutable; make tasks idempotent
- - Use Airflow or Prefect for scheduling and retries
- - Portfolio project: connected end-to-end with README and Docker

Continue Your Learning Journey

- Consolidate foundational skills first: SQL, Python, data modeling, ETL/ELT concepts
- Build one manual ETL first, then an orchestrated batch pipeline
- Add a streaming or lakehouse project to increase difficulty
- Use Kimball's Toolkit, DAMA DMBOK, and official docs for dbt, Airflow, Spark
- Practice SQL coding, data modeling whiteboards, and system design discussions
- Explore analytics engineering, platform engineering, MLOps, and streaming-first roles

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/db182dbd/public