⚙️

ETL & Data Pipelines Overview

📘Overview

A practical guide to how modern ETL/ELT data pipelines work — covering concepts, architecture, patterns, and examples using Pandas, Spark, and Airflow. Ideal for learners and engineers who want clarity without unnecessary complexity.

🧠 Structure

1. ETL Basics
2. ETL vs ELT
3. Batch vs Streaming
4. ETL Architecture
5. Data Quality & Metadata
6. ETL Patterns
7. Tools Comparison
8. Pandas/Spark/Airflow Examples
9. Real-World Pipeline Example

---

1. ETL Foundations (Extract → Transform → Load)

Extract

Extract data from databases, APIs, files, or streams. This step ensures reliable ingestion using connectors, batching, or incremental pulls.

Transform

Clean, validate, and enrich data. Includes datatype fixes, removing duplicates, applying business rules, and performing aggregations.

Load

Write transformed data into warehouses, lakes, or dashboards using full loads, incremental loads, or upserts. Ensures optimized storage for analytics.

---

2. ETL vs ELT (Modern Cloud Approach)

ETL (Traditional)

Transform before loading, used in older on-prem workflows. Reduces warehouse usage but requires heavy ETL servers.

ELT (Modern Cloud)

Load data first, then transform inside warehouses like Snowflake or BigQuery. This is faster, scalable, and more flexible for analytics teams.

---

3. Batch vs Streaming Pipelines

Batch Pipelines

Run at scheduled intervals (hourly/daily) for reports and aggregations. Best for structured workloads like sales summaries.

Streaming Pipelines

Process data continuously from sources like Kafka or Kinesis. Used for real-time analytics, fraud detection, and live dashboards.

---

4. Production ETL Architecture (Simplified)

Source Layer

Where raw data originates — DBs, APIs, S3, Kafka, etc.

Ingestion Layer

Tools or scripts that pull data, manage retries, and handle incremental loads.

Raw Layer (Bronze)

Stores unmodified data for auditability and reprocessing.

Transform Layer (Silver)

Applies cleaning and business rules using Spark, dbt or SQL.

Curated Layer (Gold)

Analytics-ready tables for BI, ML, or reporting.

Orchestration Layer

Airflow/Prefect/Dagster schedule and monitor pipeline execution.

Observability Layer

Logging, metrics, and alerts to detect failures or delays.

---

5. Data Quality, Observability & Metadata

Data Quality Checks

Ensures accuracy and trust. Includes null checks, schema validation, duplicate detection, and allowed-range checks.

Metadata & Lineage

Tracks where data came from, how it was transformed, and which jobs produced it. Helps debugging and compliance.

Monitoring & Alerts

Detects late data, failed DAGs, missing partitions, or quality failures.

---

6. Essential ETL Patterns

Incremental Loads

Process only new or changed data using timestamps, watermarks, or CDC to reduce cost and latency.

Idempotency

Pipelines should produce the same output even if re-run. Prevents duplicates and inconsistent results.

Partitioning Strategy

Partition data by date, region, or entity to improve query speed and parallel processing.

SCD (Slowly Changing Dimensions)

Manages historical data:

- Type 1: overwrite changes
- Type 2: preserve history
- Type 3: store limited history

Checkpointing (Streaming)

Stores progress so pipelines can resume processing after failure without data loss.

---

7. Tools Comparison (Quick View)

Orchestration

- **Airflow** → mature, reliable, widely adopted for scheduled pipelines.
- **Prefect** → simpler, Pythonic, great for cloud-native flows.
- **Dagster** → metadata-driven, strong asset-level workflows.

Processing

- **Spark** → large-scale distributed compute for TB-level data.
- **Dask** → parallel compute on local clusters for medium workloads.
- **Pandas** → fast, simple, in-memory processing for small data.

Transformations (SQL-focused)

- **dbt** → versioned SQL models, testing, lineage — ideal for ELT in warehouses.

---

8. ETL Examples (Pandas, Spark, Airflow)

Pandas (Small Data)

Use for quick local transformations or prototyping.

```
df = pd.read_csv("sales.csv")
df.drop_duplicates(inplace=True)
df["revenue"] = df["qty"] * df["price"]
df.to_csv("clean_sales.csv", index=False)
```

Spark (Big Data)

Handles distributed compute for large datasets.

```
df = spark.read.csv("s3://raw/sales/", header=True)
df = df.dropDuplicates().withColumn("revenue", col("qty") * col("price"))
df.write.parquet("s3://clean/sales/")
```

Airflow (Orchestration)

Defines task dependencies and scheduling.

```
t1 >> t2 >> t3
```

---

9. Real-World Pipeline Example

E-commerce Daily Sales Pipeline

- Pull orders from PostgreSQL + S3 drops.
- Store raw files in S3 with date-based folders.
- Run Spark job to clean, aggregate, and enrich.
- Use dbt to build analytics tables with SCD Type 2.
- Load into Snowflake for BI dashboards.
- Airflow orchestrates extract → transform → load → quality checks.
- Alerts fire on missing partitions or failed jobs.

🎯 Takeaway

This cheatsheet gives a clean, practical view of ETL & Data Pipelines — covering concepts, architecture, tools, and real examples without overwhelming detail.
