DATA ENGINEERING | INTERMEDIATE | 3 DAYS

Python for Data Engineering

Build a strong foundation in building production-style data pipelines using Python, from ingestion to incremental processing and data quality checks. Learn how to design maintainable pipelines with logging, retries, validations, and reporting patterns used in real-world batch processing systems.

Why Python DE?

Faster pipeline development

Build ingestion, transformation, and automation quickly with Python ecosystems.

Enables advanced use cases

Python unlocks ML, NLP, and automation workflows beyond SQL.

Better integration support

Connect APIs, databases, and cloud services with mature libraries.

Higher team productivity

Engineers deliver more with reusable scripts and modular pipeline code.

Improved data quality automation

Validate, profile, and test data programmatically at scale.

What you'll be able to do

- ✓ Use Python confidently for day-to-day data engineering tasks.
- ✓ Build reliable ingestion and transformation scripts for batch pipelines.
- ✓ Work with common pipeline formats including CSV, JSON, and Parquet (intro).
- ✓ Connect Python pipelines to databases and data warehouses.
- ✓ Implement robust transformations using pandas and hybrid SQL + Python approaches.
- ✓ Build incremental ingestion logic using watermark loads and deduplication.
- ✓ Handle schema changes, bad records, and late-arriving data safely.
- ✓ Implement data quality checks and validations in Python.
- ✓ Build modular, config-driven, production-ready pipeline code with logging, error handling, and retry patterns.
- ✓ Deliver an end-to-end mini data pipeline as a capstone project.

Who this is for

- Beginner to intermediate Data Engineers
- Data Analysts moving into Data Engineering
- Backend engineers working with data pipelines
- Analytics engineers using Python + SQL
- Anyone building batch pipelines and data automation scripts

Curriculum

01 Python for Data Engineering (Core Skills and Workflow)

- Python essentials for pipelines (types, functions, modules)
- Virtual environments and dependency management basics
- Project structure for data engineering scripts (src, configs, tests)
- Working with time and timestamps for pipeline runs
- **Hands-on:** Lab: Build a simple pipeline skeleton that reads input, processes, and writes output with proper exit codes

02 Building Batch Ingestion Scripts (Reliable Data Loads)

- Ingestion patterns (file drop, API pull, database extract concepts)
- File discovery and run-by-date processing (partition folders)
- Idempotency basics (safe re-runs, overwrite vs append)
- Writing staging outputs and run metadata
- **Hands-on:** Lab: Build a batch ingestion script that loads daily files into a staging area

03 Working with CSV and JSON (Parsing, Schemas, Pitfalls)

- Reading CSV safely (dtypes, delimiters, encoding, quoting)
- Schema definition vs inference and why it matters
- Working with JSON (nested fields, flattening, normalization)
- Handling corrupt records and bad rows (quarantine pattern concept)
- **Hands-on:** Lab: Ingest messy CSV + nested JSON, normalize fields, and write clean outputs

04 Parquet Introduction (Columnar Storage for Pipelines)

- Why Parquet (compression, column pruning, analytics performance)
- Writing Parquet with stable schemas (types and nullability)
- Partitioned Parquet outputs (date, region, source)
- Small files awareness and file sizing basics
- **Hands-on:** Lab: Convert cleansed CSV/JSON data into partitioned Parquet outputs

05 Connecting to Databases and Warehouses (Extraction and Loading)

- DB connectivity basics (connection strings, pooling concepts)
- Reading with SQL and loading into pandas for transformations
- Writing results back to PostgreSQL (upsert concepts)
- Warehouse connectivity concepts (drivers and auth patterns overview)
- **Hands-on:** Lab: Extract data from PostgreSQL, transform in Python, and load curated tables back

06 Transformations with pandas and Hybrid SQL + Python

- Core pandas transformations (filter, select, merge, groupby)
- Date/time transformation patterns (standardize, timezone awareness concept)
- Hybrid approach: do heavy joins/filters in SQL, refine in pandas
- Performance basics (avoid row-wise loops, vectorization patterns)
- **Hands-on:** Lab: Build a transformation step that uses SQL extraction + pandas enrichment + curated output

07 Incremental Loads (Watermarks and Deduplication)

- Full refresh vs incremental and when to choose each
- Watermark loads (updated_at, ingestion timestamp) and storing state
- Deduplication patterns (business keys, last-write-wins concepts)
- Backfill and reprocessing windows for missed days
- **Hands-on:** Lab: Implement incremental ingestion with a watermark and validate multiple runs

08 Handling Schema Changes, Bad Records, Late-Arriving Data

- Schema drift detection (new columns, type changes) and safe handling
- Bad records strategy (drop, quarantine, fix-forward workflows)
- Late-arriving data patterns (reprocess window, merge concepts)
- Idempotent processing for retries and partial failures
- **Hands-on:** Lab: Simulate schema changes and corrupt records and implement safe pipeline behavior

09 Data Quality Checks and Validation in Python

- Validation patterns (nulls, duplicates, ranges, referential checks concepts)
- Fail-fast vs warn-only design and when to use each
- Generating validation reports (counts, failed rows, summary)
- Quality gates before publishing curated outputs
- **Hands-on:** Lab: Add a validation stage that blocks publishing and outputs a quality report

10 Production-Ready Pipeline Code (Config-Driven, Logging, Retries)

- Config-driven pipelines (YAML/JSON + env vars + CLI args)
- Structured logging (run_id, dataset, row counts, timing)
- Error handling strategy (categorize errors, retries with backoff concepts)
- Modular design (extract, transform, load layers) and testing basics
- **Hands-on:** Lab: Refactor scripts into modular pipeline code with configs, logging, and retry behavior

11 Capstone Project (End-to-End Mini Data Pipeline)

- Capstone goal: Deliver a working batch pipeline with quality checks
- Ingest CSV/JSON sources and produce partitioned Parquet staging
- Transform using pandas + SQL and load curated tables into PostgreSQL
- Implement incremental loads with watermark and deduplication
- Add schema drift handling, bad record quarantine, and validation report
- **Hands-on:** Capstone Lab: Deliver runnable pipeline code, configs, logs, and sample outputs with a short runbook