Tribhuvan University
Institute of Engineering
Purwanchal Campus
पूर्वाञ्चल क्याम्पस

Unit 6
Data Engineering and Automation

Contents

Contents

Data Engineering

- Data engineering is the discipline that designs, builds, and maintains systems for collecting, storing, and processing data so it can be used for analysis, machine learning, and decision making.
- It focuses on:
  - data flow
  - system architecture
  - scalability
  - reliability
- It is infrastructure work, not analysis work.

Position in Data Science Ecosystem

- Typical ecosystem roles:
  - Data Engineer → prepares data
  - Data Scientist → analyzes and models
  - Data Analyst → interprets results
  - ML Engineer → deploys models
- Relationship:

  Raw Data → Data Engineering→ Prepared Data→ Data Science
- Without this stage, analysis quality drops.

Goals of Data Engineering

- Accessibility
  - Data available when needed
- Consistency
  - Uniform format across sources
- Scalability
  - Handles growing volume
- Reliability
  - Pipelines run without failure
- Efficiency
  - Optimized resource use

Data Lifecycle Perspective

- Data engineering covers major stages of lifecycle:
  - Generation
  - Collection
  - Storage
  - Processing
  - Distribution
  - Archival or deletion

Data Storage Paradigms

- Relational storage
  - Usually structured tables which are schema enforced & SQL based
- NoSQL storage
  - Flexible schema stored as a document or key-value & scalable horizontally
- Data lakes
  - raw storage with large volume support
- Data warehouses
  - optimized for analytics
- Choice depends on workload.

Skills and Knowledge Areas

- Technical:
  - Programming
  - Database systems
  - Distributed computing
  - Networking basics
- Conceptual:
  - System design
  - Abstraction
  - Problem decomposition
-

Challenges in Practice

- Heterogeneous data sources
- Evolving schemas
- System bottlenecks
- Cost control
- Data privacy laws

Contents

ETL

- ETL (Extract, Transform, Load) is a structured process used to move data from source systems into storage systems prepared for analysis.
- It involves:
  - Extract — collecting raw data
  - Transform — cleaning and shaping
  - Load — storing in target system
- ETL ensures integration, consistency, and usability of data.

Purpose of ETL

- Combine multiple data sources
- Improve data quality
- Standardize formats
- Support analytics and reporting
- Enable historical storage
- ETL is a core mechanism in data warehouses and pipelines.

Extraction

- Extraction retrieves data from sources:
  - relational databases, APIs, logs, flat files, sensors etc.
- Challenges:
  - different formats
  - network latency
  - incomplete data
  - permission control

Methods for Extraction

- Full extraction
  - entire dataset copied
  - Simple but high resource cost
- Incremental extraction
  - only changed records
  - Efficient but requires tracking
- Change Data Capture (CDC)
  - detects real-time changes
  - used in modern pipelines

Transformation

- Transformation prepares data for use.
- Typical operations:
  - Cleaning
  - Normalization
  - Aggregation
  - Joining datasets
  - Filtering
  - Feature creation

Loading

- Loading transfers processed data to target system.
- Targets:
  - Databases
  - Warehouses
  - Data lakes
- Objectives:
  - Fast insertion
  - Integrity preservation
  - Indexing support

Loading Strategies

- Full load
  - replaces data
  - simple approach
- Incremental load
  - adds updates only
  - efficient
- Upsert strategy
  - update or insert
  - maintains current state

Contents

Workflow Automation

- Workflow automation means running tasks automatically at planned times or events.
- In data pipelines this includes:
  - running ETL jobs
  - updating datasets
  - generating reports
  - backups
- Automation reduces manual work and errors.

Importance of Scheduling

- Ensures regular execution
- Maintains data freshness
- Supports reproducibility
- Reduces human intervention
- Improves reliability
- Without scheduling, pipelines depend on manual triggers.

Types of Scheduling

- Time-based
  - run at fixed times
- Event-based
  - run when condition met
- Dependency-based
  - run after another task finishes
- Modern systems often combine all three.

Introduction to CRON

- CRON is a time-based scheduler used in Unix/Linux systems.
- It executes commands automatically.
- Used for:
  - scripts
  - backups
  - ETL jobs
- Runs in background as a system service.

Introduction to CRON

- Cron is a software utility used to schedule jobs (commands or shell scripts) to run periodically at fixed times, dates, or intervals.
- **Cron Job:** A specific task scheduled to run automatically, such as daily backups or system maintenance.
- **Crontab:** The configuration file (cron table) where users define their schedules.

CRON Time Format

- A standard cron expression consists of five fields:

  minute hour day_of_month month day_of_week

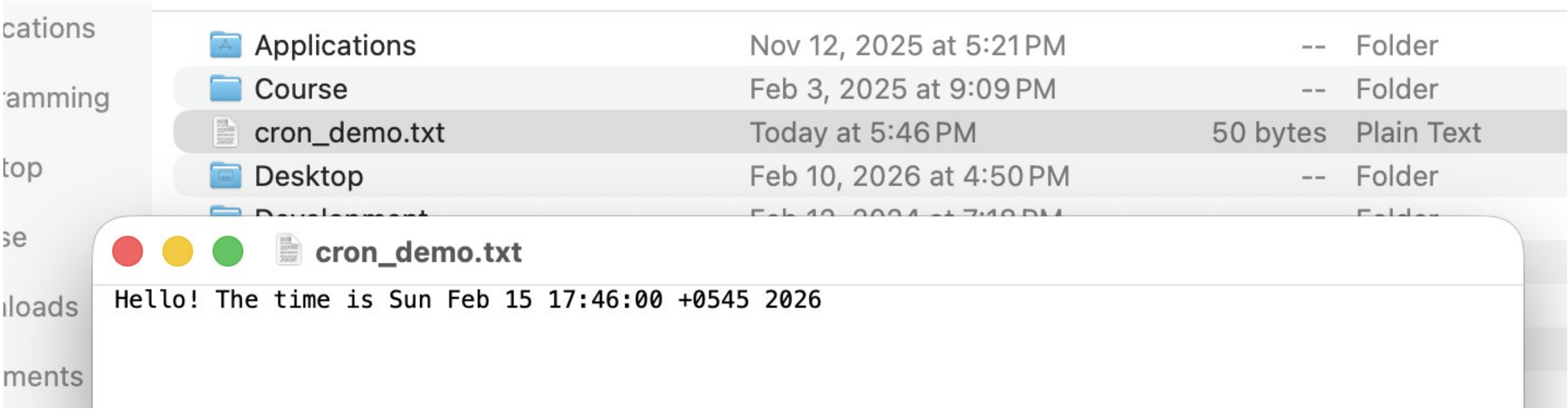
- Example pattern:
  - 0 0 * * * runs a command every night at midnight.
  - 30 2 * * * schedules a task to run at 2:30 AM every day.

Example

Example

nano
⌥⌘1
UW PICO 5.09
File: /tmp/crontab.m9cxLkMgPp
Modified
* * * * * echo "Hello! The time is $(date)" > /Users/pukarkarki/cron_demo.txt

Example


cations
amming
top
se
loads
ments
Applications
Nov 12, 2025 at 5:21 PM
--
Folder
Course
Feb 3, 2025 at 9:09 PM
--
Folder
cron_demo.txt
Today at 5:46 PM
50 bytes
Plain Text
Desktop
Feb 10, 2026 at 4:50 PM
--
Folder
cron_demo.txt
Hello! The time is Sun Feb 15 17:46:00 +0545 2026

Example

```
-zsh                                                                 ⌥⌘1
pukarkarki@macbookpro ~ % crontab -l
* * * * * echo "Hello! The time is $(date)" > /Users/pukarkarki/cron_demo.txt
```

Real Life Examples of CRON

- **E-Commerce: Nightly Inventory Sync**
  - Goal: Sync stock levels between a physical warehouse and an online store.
  - Schedule: 0 0 * * * (Every night at midnight).
- **FinTech: Monthly Invoicing**
  - Goal: Generate and email PDF invoices to all active subscribers.
  - Schedule: 0 8 1 * * (8:00 AM on the 1st of every month).
- **Marketing: Daily Performance Reports**
  - Goal: Scrape website analytics and email a summary to the marketing team.
  - Schedule: 30 7 * * 1-5 (7:30 AM every weekday).

Advantages/Limitations of CRON

- Advantages
  - Lightweight
  - Reliable
  - Widely available
  - Simple configuration
- Limitations:
  - Limited dependency control
  - Basic monitoring
  - Manual logging setup needed

Python schedule Library

- **schedule** is a Python tool for simple job automation.
- Runs inside Python programs.
- Features:
  - readable syntax
  - easy integration
  - flexible intervals
- Best for small to medium tasks.

Python schedule Library

```
(AI) pukarkarki@macbookpro ~ % pip3 install schedule
Collecting schedule
  Downloading schedule-1.2.2-py3-none-any.whl.metadata (3.8 kB)
Downloading schedule-1.2.2-py3-none-any.whl (12 kB)
Installing collected packages: schedule
Successfully installed schedule-1.2.2
```

automate.py > ...

```python
import schedule
import time
from datetime import datetime

def write_file():
    with open("schedule_demo.txt", "w") as f:
        f.write(f"Hello! The time is {datetime.now()}")

schedule.every(1).minutes.do(write_file)

while True:
    schedule.run_pending()
    time.sleep(1)

```

Python schedule Library

```
schedule.every(5).minutes.do(job)
schedule.every().hour.do(job)
schedule.every().day.at("14:30").do(job)
schedule.every().monday.do(job)
```

System Scheduler vs Application Scheduler

- System-level
  - CRON
  - independent of program
  - stable for servers
- Application-level
  - schedule library
  - embedded in code
  - flexible logic

Contents

Logging, Monitoring, and Error Handling in Pipelines

- Data pipelines run automatically and often without direct supervision.
- To maintain reliability, systems must:
  - record activity
  - observe performance
  - handle failures
- This is achieved through:
  - Logging
  - Monitoring
  - Error handling
- These ensure stable pipeline operation.

Importance in Data Engineering

- Logging, Monitoring & Error handling are critical because
  - pipelines process large data
  - failures can go unnoticed
  - incorrect data affects analysis
  - downtime impacts decisions
- They provide visibility and control over pipeline behavior.

Logging

- Logging is the recording of system events during execution.
- Logs contain:
  - timestamps
  - task status
  - messages
  - error details

Types of Logs

- Execution logs
  - track job progress
- Error logs
  - record failures
- Audit logs
  - track user/system actions
- Performance logs
  - capture timing and usage

Logging Levels

- Standard severity levels:
  - DEBUG — detailed internal info
  - INFO — normal operation
  - WARNING — potential issues
  - ERROR — failures occurred
  - CRITICAL — system stopping issues

Logging Best Practices

- Use structured messages
- Include timestamps
- Log meaningful context
- Avoid excessive verbosity
- Store logs centrally

Good logs simplify debugging.

Monitoring

- Monitoring is continuous observation of system health and performance.
- It answers:
  - Is pipeline running?
  - How fast?
  - Any failures?
- Provides real-time awareness.

Monitoring Metrics

- Common metrics:
  - Execution time
  - Success rate
  - Throughput
  - Resource usage
  - Queue length
- Metrics indicate pipeline efficiency.

Monitoring Methods

- Dashboards
- Automated alerts
- Periodic checks
- Visual analytics

Error Handling

- Error handling is the structured response to unexpected issues during execution.
- Goals:
  - prevent crashes
  - protect data integrity
  - recover gracefully
- This maintains pipeline continuity.

Types of Errors

- Syntax errors
  - Coding mistakes
- Runtime errors
  - Occur during execution
- Logical errors
  - Incorrect results
- System errors
  - Hardware/network failure

Techniques for Error Handling

- Try–except structures
- Retries
- Fallback steps
- Checkpoint recovery
- Transaction rollback

Fault Tolerance

- Fault tolerance means continuing operation despite failures.
- Achieved through:
  - Redundancy
  - Replication
  - Distributed execution
  - Checkpoint storage
- Important for large-scale systems.

Contents

Data Storage and Retrieval Strategies for Pipelines

- Data pipelines generate and process large volumes of data.
- Efficient storage and retrieval strategies ensure:
  - fast access
  - reliability
  - scalability
  - cost control
- This stage determines overall pipeline performance.

Goals of Storage Strategy

- Preserve data integrity
- Support efficient querying
- Enable scalability
- Ensure availability
- Optimize cost

Different Storage Architecture

- Data moves through layers:

  Raw Storage → Processed Storage → Analytical Storage

- Each layer serves a different purpose and format.

Types of Storage

- Relational Databases
  - structured tables
  - strong consistency
- NoSQL Databases
  - flexible schema
  - high scalability
- Data Warehouses
  - analytics optimized

Relational Storage

- Schema-based structure
- SQL querying
- Transactional integrity
- Normalization support
- Best suited for structured data and reporting.
- Limitations:
  - less flexible scaling
  - rigid schema changes

NoSQL Storage

- Types:
  - document stores
  - key-value stores
  - column-based
  - graph databases
- Advantages:
  - flexible schema
  - horizontal scaling
- Trade-off:
  - weaker consistency guarantees

Date Warehouse

- Designed for analytics.
- Features:
  - integrated data sources
  - historical storage
  - optimized aggregation queries
  - columnar storage models
- Supports decision making and reporting.

Contents

Automated Report Generation

- Automated report generation is the process of producing formatted outputs from processed data without manual intervention.
- Reports communicate results to users in readable form.
- Common formats:
  - Excel
  - HTML
  - PDF
- This is usually the final stage of a pipeline.

Automated Report Generation

- Why?
  - Deliver insights regularly
  - Reduce manual work
  - Ensure consistency
  - Improve decision support
  - Support large-scale distribution

Reporting Formats

- Excel
  - Interactive tables
  - User manipulation possible
- HTML
  - Web display
  - Accessible remotely
- PDF
  - Fixed layout
  - Professional sharing
- Choice depends on audience needs.

Automation Workflow

- **Steps:**

1) Data analysis completed
2) Format template selected
3) Content inserted
4) Report exported
5) Delivered to users

- Reports often generated daily, weekly or monthly and schedulers trigger generation automatically after pipeline completion.

Contents

Q) Design an end-to-end automated analytics pipeline for a real-world organization (such as retail, hospital, banking, or education). Explain:

- data sources
- ETL process
- storage choice
- scheduling method
- logging and monitoring
- report generation