Road map to Data Engineer

Contents

[File Handling & Data Formats 2](#_Toc202535421)

[Objectives: 2](#_Toc202535422)

[Key Libraries: 2](#_Toc202535423)

[Practice Tasks: 2](#_Toc202535424)

[Data Cleaning & Formatting 2](#_Toc202535425)

[Objectives: 2](#_Toc202535426)

[Key Functions: 2](#_Toc202535427)

[Sample Tasks: 3](#_Toc202535428)

[Exploratory Data Analysis (EDA) 3](#_Toc202535429)

[Objectives: 3](#_Toc202535430)

[Tools: 3](#_Toc202535431)

[Example Projects: 3](#_Toc202535432)

[Data Aggregation & Transformation 3](#_Toc202535433)

[Concepts: 3](#_Toc202535434)

[Tasks: 3](#_Toc202535435)

[Mini Projects for Confidence 3](#_Toc202535436)

[Stretch Goals 3](#_Toc202535437)

[Logging & Monitoring 4](#_Toc202535438)

[Objectives: 4](#_Toc202535439)

[Key Libraries: 4](#_Toc202535440)

[Practice Tasks: 4](#_Toc202535441)

[Orchestration & Scheduling 4](#_Toc202535442)

[Objectives: 4](#_Toc202535443)

[Tools: 4](#_Toc202535444)

[Sample Tasks: 4](#_Toc202535445)

[Cloud Storage Integration 4](#_Toc202535446)

[Objectives: 4](#_Toc202535447)

[Tools: 4](#_Toc202535448)

[Sample Tasks: 4](#_Toc202535449)

[Databases & SQL Connectivity 5](#_Toc202535450)

[Objectives: 5](#_Toc202535451)

[Key Libraries: 5](#_Toc202535452)

[Practice Tasks: 5](#_Toc202535453)

[Week 1: File Formats & Fundamentals (Jul 4–10) 5](#_Toc202535454)

[Week 2: Data Cleaning & Formatting (Jul 11–17) 5](#_Toc202535455)

[Week 3: EDA + Transformations (Jul 18–24) 5](#_Toc202535456)

[Week 4: Feature Engineering + Mini Pipelines (Jul 25–31) 6](#_Toc202535457)

[Week 5: Production-Ready Pipelines & Systems (Aug 1–7) 6](#_Toc202535458)

# File Handling & Data Formats

## Objectives:

- Understand and work with CSV, Excel, JSON, XML, and Parquet

- Handle encodings, delimiters, nested data

## Key Libraries:

- `pandas`, `openpyxl`, `xlrd`, `json`, `xml.etree.ElementTree`, `lxml`, `pyarrow`

## Practice Tasks:

- Parse nested JSON and flatten it

- Fix malformed CSV with unknown encoding

- Convert Excel sheets to CSV

# Data Cleaning & Formatting

## Objectives:

- Deal with missing, duplicate, and inconsistent data

- Standardize formats: dates, categories, text casing

- Perform outlier detection and basic validations

## Key Functions:

- `dropna`, `fillna`, `duplicated`, `replace`, `apply`

- `str.replace`, `re.sub`, `pd.to\_datetime`, `fuzzywuzzy`

## Sample Tasks:

- Clean scraped product listings

- Normalize addresses with fuzzy matching

- Format date columns consistently

# Exploratory Data Analysis (EDA)

## Objectives:

- Visualize distributions and relationships

- Identify trends, anomalies, and correlations

## Tools:

- `pandas`, `matplotlib`, `seaborn`, `plotly`

## Example Projects:

- Analyze irrigation data by region

- Create heatmaps and box plots

- Compare schemes across districts interactively

# Data Aggregation & Transformation

## Concepts:

- `groupby`, `pivot\_table`, `melt`, `merge`, `concat`

- Time-based operations: `.resample()`, `.rolling()`, `.expanding()`

## Tasks:

- Aggregate census data by state

- Find top-performing irrigation schemes by village

- Generate rolling averages for groundwater levels

# Mini Projects for Confidence

- Village Census Analyzer – Clean and visualize irrigation statistics

- Government Scheme Breakdown – Transform and compare scheme coverage

- Universal Dataset Formatter – Accept any messy input, return clean and structured output

# Stretch Goals

- Use `pydantic` or `jsonschema` for schema validation

- Save data with compression (gzip, Parquet)

- Build parameterized ETL scripts with config files (`argparse`, `.yaml`, `.env`)

# Logging & Monitoring

## Objectives:

Implement structured logs for traceability and debugging

Track pipeline behavior and capture failures gracefully

## Key Libraries:

logging, logging.config, python-json-logger, loguru

## Practice Tasks:

Integrate structured logging in a data cleaning script

Tag each step in an ETL pipeline with info/warning/error levels

Log bad records to a separate file with clear error reasons

# Orchestration & Scheduling

## Objectives:

Automate multi-step pipelines with dependencies and retries

Schedule jobs, monitor workflows, and manage tasks dynamically

## Tools:

Apache Airflow, Prefect, Dagster

## Sample Tasks:

Set up a DAG to fetch, clean, and store daily rainfall data

Add retry logic on API pull with exponential backoff

Use task parameters to make a dynamic irrigation scheme ETL pipeline

# Cloud Storage Integration

## Objectives:

Read/write data from cloud buckets to enable scalable workflows

Store artifacts in compressed formats for faster data flow

## Tools:

boto3 (AWS S3), gcsfs (GCP), s3fs, smart\_open

File formats: .gz, .snappy, .parquet

## Sample Tasks:

Upload transformed village census data to S3 as Parquet

Build a function that reads any CSV from GCS and logs metadata

Compare read speeds: local CSV vs gzipped Parquet in cloud

# Databases & SQL Connectivity

## Objectives:

Store structured data in relational databases

Query large datasets directly from your scripts

## Key Libraries:

sqlalchemy, psycopg2, sqlite3, pandas.read\_sql()

## Practice Tasks:

Create PostgreSQL tables for irrigation records

Load scheme metadata into SQLite and perform joins via pandas

Write a function that writes cleaned records to a local DB

# Week 1: File Formats & Fundamentals (Jul 4–10)

| **Day** | **Focus Area** | **Tasks / Goals** |
| --- | --- | --- |
| 1 | CSV/TSV Basics | Load, clean, handle encoding & delimiters |
| 2 | Excel Sheets | Read multiple sheets, clean data types |
| 3 | JSON Files | Flatten nested JSON, extract specific keys |
| 4 | XML Handling | Parse and navigate XML trees |
| 5 | Parquet & Compressed Data | Write/read .parquet, gzip, and .zip files |
| 6 | Format Conversion | Convert across formats: JSON ⇄ CSV, Excel ⇄ CSV |
| 7 | Mini Challenge | Clean and combine mixed-format data into a standard CSV |

# Week 2: Data Cleaning & Formatting (Jul 11–17)

| **Day** | **Focus Area** | **Tasks / Goals** |
| --- | --- | --- |
| 8 | Missing/Duplicates | Use dropna, fillna, duplicated, null logic |
| 9 | Text Cleaning | Regex, str.lower, remove special characters |
| 10 | Date/Time Standardization | Convert formats, timezone handling |
| 11 | Column Normalization | Rename, reorder, infer types |
| 12 | Outlier Detection | Boxplot, Z-score, IQR method |
| 13 | Validation Rules | Build checks for valid input ranges, schema design |
| 14 | Challenge Day | Clean a real dataset: irrigation census or similar |

# Week 3: EDA + Transformations (Jul 18–24)

| **Day** | **Focus Area** | **Tasks / Goals** |
| --- | --- | --- |
| 15 | Descriptive Stats | describe(), distributions, quantiles |
| 16 | Grouping & Aggregation | groupby, pivot\_table, basic analysis |
| 17 | Joins/Merges | Combine tables with merge, concat |
| 18 | Categorical Analysis | Frequency counts, scheme codes |
| 19 | Time-Series | Resampling, rolling averages |
| 20 | Visualization | Line plots, histograms, heatmaps (matplotlib/seaborn) |
| 21 | EDA Project | Full exploratory report with visuals + summary |

# Week 4: Feature Engineering + Mini Pipelines (Jul 25–31)

| **Day** | **Focus Area** | **Tasks / Goals** |
| --- | --- | --- |
| 22 | Feature Creation | Derived columns, ratios, flags |
| 23 | Encoding Techniques | One-hot, label encoding, mapping schemes |
| 24 | Configurable ETL Script | Use argparse or config files to parameterize ETL runs |
| 25 | Schema Validation | Use pydantic or manual schema rules |
| 26 | Compression & Saving | Write to compressed formats: .zip, .gzip, .parquet |
| 27 | Logging & Error Handling | Add logging, graceful failure points |
| 28 | Final Project | A complete ETL app that ingests, cleans, analyzes, and outputs |

# Week 5: GCP-Ready Production Pipelines (Aug 1–7)

| **📅 Day** | **Focus Area** | **Goals & Practice** | **Key GCP Tools / Concepts** |
| --- | --- | --- | --- |
| **29** | 🪣 Cloud Storage & File Operations | Use gcsfs with Pandas to read/write from GCS buckets. Practice file conversion (CSV ↔ Parquet), versioning, folder structuring. | Cloud Storage, gcsfs, fsspec, IAM roles |
| **30** | 🧠 Cloud-Based Data Cleaning | Ingest raw files from GCS, clean and format in Prefect/Colab/Vertex notebooks, and save cleaned files back. | Cloud Storage, Prefect, Vertex Workbench, pandas, Cloud Functions (optional trigger) |
| **31** | 🔍 BigQuery Connectivity | Load transformed data into BigQuery via pandas\_gbq. Perform joins with other tables. Learn partitioning + clustering. | BigQuery, pandas\_gbq, google-cloud-bigquery, SQLAlchemy (optional) |
| **32** | 🔔 Pipeline Monitoring & Alerts | Add structured logging via python-json-logger. Save failed records to GCS or log table in BigQuery. Simulate alert via Pub/Sub or email. | Logging, Pub/Sub, Cloud Monitoring, BigQuery audit table |
| **33** | 🛠️ Orchestration & Scheduling | Build and deploy a Prefect Flow: GCS → clean → transform → log → BigQuery. Schedule runs via Prefect Cloud or GCP Scheduler. | Prefect, GCS, BigQuery, Cloud Scheduler, Cloud Run (optional) |
| **34** | 🧪 Refactor Final Project | Repackage your ETL project to use GCP components. Add config files, CLI args, secrets. Make it modular, parameterized, and cloud-ready. | Prefect, GCS, BigQuery, .env, YAML/config loader |
| **35** | 📦 Show & Tell Day | Publish your project: clean repo with README + sample data + screenshots. Bonus: deploy as a scheduled app via Prefect Cloud or Cloud Run. | GitHub, GCS, BigQuery, Prefect Cloud UI |

## Optional Extras (Pick 1–2 to Level Up)

* Add **Cloud Function trigger** to auto-launch ETL when new file lands
* Use **Terraform** to provision buckets, datasets, IAM roles
* Embed a **Streamlit dashboard** that reads GCS/BigQuery data
* Track metadata with a **BigQuery audit log table**: timestamp, row counts, errors