**🔹 End-to-End Data Engineering Project — Step by Step**

**1. Set up your environment**

* Install: Python, Docker, Airflow (or Prefect), Spark (PySpark), dbt, Postgres (or Azure SQL), and Azure CLI.
* Use **Docker Compose** to spin up Postgres (for local testing) and Airflow (to orchestrate tasks).
* Create a Git repo with folders: ingest/, spark/, transform/, dbt/, dags/.

**2. Build the ETL pipeline (API → Database)**

* Identify an open API (e.g., orders, weather, or finance data).
* Write a small Python script to **fetch data from the API**, save it as CSV or JSON in a “raw” folder.
* Load the raw data into Postgres (or Azure SQL Database) as a **staging table**.
* Keep your ingestion **idempotent** (re-runs should not duplicate data).

**Deliverable:** A repeatable API ingestion process that loads staging tables in the DB.

**3. Process a large dataset with Spark**

* Get a **large CSV dataset** (millions of rows) from Kaggle or open data portals.
* Use PySpark to:
  + Clean missing values
  + Convert datatypes (e.g., date formats, numeric fields)
  + Deduplicate records
  + Partition by time or category for faster queries
* Save the processed data as **Parquet files** in a “staging” zone (locally or on Azure Blob/ADLS).

**Deliverable:** A Spark pipeline that processes large raw files → cleansed, partitioned Parquet in staging.

**4. Transform data into a star schema with dbt**

* Define a **star schema**:
  + Fact table (e.g., fact\_sales)
  + Dimension tables (dim\_customer, dim\_product, dim\_date)
* In dbt:
  + Create **staging models** from your raw/staging tables
  + Build **fact + dimension models**
  + Add **tests** (check for duplicates, nulls, referential integrity)
* Run dbt locally against your Postgres database.

**Deliverable:** A warehouse layer with fact + dimension tables and tests.

**5. Orchestrate the workflow (Airflow or Prefect)**

* Define tasks:
  1. Ingest API data
  2. Run Spark job
  3. Transform/load into DB (fact + dims)
  4. Run dbt models/tests
* Chain the tasks into a DAG (Airflow) or Flow (Prefect).
* Schedule it daily/weekly depending on your dataset.
* Add logging and retries for robustness.

**Deliverable:** A single DAG/Flow that automates the pipeline end to end.

**6. Containerize & document**

* Create Dockerfiles for your scripts (so they run consistently everywhere).
* Use docker-compose to run Postgres, Airflow, and your ETL code locally.
* Document the project: architecture diagram, folder structure, how to run.
* Push everything to GitHub (with clear README).

**Deliverable:** Reproducible environment anyone can run with docker-compose up.

**7. Deploy to Azure (optional for bonus)**

* Replace local Postgres with **Azure SQL Database**.
* Replace local storage with **Azure Blob Storage / ADLS**.
* Run Spark jobs on **Azure Databricks**.
* Run Airflow on **Azure Kubernetes Service** or use **Prefect Cloud**.
* Keep secrets in **Azure Key Vault**.

**Deliverable:** Cloud-deployed pipeline running on Azure.

**🔹 What You’ll End Up With**

1. **ETL pipeline** → from API → raw → DB.
2. **Data warehouse (star schema)** → fact + dims built with dbt.
3. **Big data pipeline** → Spark processes large datasets efficiently.
4. **Automation** → Airflow/Prefect DAG orchestrates everything.
5. **Production readiness** → Dockerized, documented, testable.

Airflow, Prefect, PySpark, Pandas, SQLAlchemy, DBT, Docker, Azure.

1. ETL pipeline API → DB.

2. Data warehouse with star schema.

3. Spark pipeline on large dataset. with a project step by step