What I learnt from the course:
I did a project in Fabrics for my final project. Following are my learnings:

- **Building and Maintaining Data Pipelines**: Designed and implemented robust ETL pipelines using Microsoft Fabric and Apache Spark to integrate multi-source hydropower data from Fingrid APIs, Zenodo, and JRC datasets. Automated ingestion, transformation, and aggregation workflows enabled seamless transition from raw to curated data layers, ensuring consistent refreshes and data lineage across the lakehouse environment.
- **Data Infrastructure:** Developed a scalable lakehouse architecture leveraging Delta tables to unify structured and semi-structured datasets. Implemented Bronze–Silver–Gold layers to maintain modularity and transparency in data transformations, supporting both real-time and historical analytics. The architecture demonstrated best practices in cloud-native data storage and processing using Microsoft Fabric.
- **Data Processing tools**: Applied Apache Spark, PySpark, and SQL extensively for distributed data processing, aggregation, and feature computation. Used Delta Lake for transactional reliability, schema evolution, and ACID compliance. Implemented vectorized analytics through FAISS and explored integration with Azure OpenAI for semantic data retrieval.
- **Data Modeling**: Modeled datasets to capture key hydropower performance metrics such as observed capacity factor, baseline average, and deviations. Created dimensional and analytical models aligning with the Medallion architecture, enabling clear traceability between operational data, historical baselines, and analytical KPIs.
- **DevOps & CI/CD**: Adopted modular notebooks. Further, I want to expand my work by parameterizing the ETL processes that could be version-controlled and automated within Fabric pipelines. Demonstrated CI/CD-ready design principles for reproducible transformations, schema consistency, and data validation at each stage of the processing lifecycle.
- **Data Quality:** Implemented validation checks to ensure consistency between real-time and historical datasets. Addressed schema mismatches, null handling, and type-casting errors during pipeline operations. Applied aggregation and normalization techniques to ensure reliable comparison of capacity factors across multiple hydropower typologies.
- **Scalability and Performance**: Optimized Spark transformations and partitioning strategies to efficiently handle large time-series datasets. Designed the architecture to scale horizontally for multi-year hydropower data analysis while maintaining performance through Delta caching and efficient query planning.
- **Collaboration**: Enabled cross-functional usability by preparing curated gold datasets consumable by Power BI analysts, data scientists, and policy researchers. Provided semantically enriched summaries and vectorized search capability to facilitate insight discovery and decision-making across analytical teams.
- **Data Security and Governance**: Ensured compliance with organizational and regional data policies by using secure Microsoft Fabric environments and Azure OpenAI services within approved regions. Further, I would like to Follow data governance principles by

maintaining audit trails, schema control, and controlled access permissions for reproducible, transparent data operations.

With the approach I followed, the data modeling (in the *Gold layer*) followed a **dimensional/analytic approach** — optimized for reporting. I would like to further expand it by adding a **Data Vault layer (in between Silver and Gold)** that would provide:

- Full **historization** of changes (e.g., when baseline data is updated).

- Strong **auditability** and lineage for regulatory energy analytics.

- Easier **extension** when new energy sources (solar, wind) are added later.

By implementing Data Vault 2.0 design directly in Fabric using Delta tables, where each layer (hub/link/satellite) becomes a Delta dataset.

2 # dbt — Enabling CI/CD and Model Orchestration:

To **orchestrate transformations**, apply **version control**, and **test data quality** declaratively instead of managing all transformations in notebooks. dbt (Data Build Tool) would replace or complement the manual Spark SQL transformations with modular .sql models stored in a Git repository.