DTA hydraulic pump inventory

This project generates high-quality, structured hydraulic pump inventory data for advanced analytics, modeling, and decision-making. The dataset includes key attributes such as pump type, model, quantity, cost, supplier, pump individual parts and location, ensuring consistency, accuracy, and reliability. Data cleaning processes remove duplicates, standardize entries, and prepare a “gold” dataset suitable for tasks such as inventory forecasting, supplier analysis, trend detection, and operational optimization.

For this internship project, I am developing ETL pipelines on Azure using ADLS Gen2 and Databricks to manage the hydraulic pump inventory. The work encompasses designing the architecture, documenting setup procedures, establishing security and operational best practices, providing example data models, and implementing a sample pipeline ready for use.

➢ Complete Workflow Architecture:

1) High-level architecture:

- **Ingest**: Azure Data Factory (ADF) or Databricks jobs pull from sources (API, CSV uploads, jsons, DB).
- **Landing / Raw storage**: ADLS Gen2 (Hierarchical Name-space) — immutable raw files (date-stamped).
- **Transform**: Databricks (Delta Lake) for bronze -> silver -> gold transforms. Use Delta tables for ACID, MERGE, time travel.
- **Orchestration**: ADF or Databricks Jobs to orchestrate notebooks/pipelines.
- **Catalog & Governance**: Azure Purview (or Databricks Unity Catalog) for schema, lineage, and data discovery.
- **Security**: Managed Identities, Key Vault for secrets.
- **Monitoring**: Log Analytics + Azure Monitor + Databricks metrics .
- **CI/CD**: Git (feature branches), GitHub Actions / Azure DevOps pipelines, Databricks CLI or workspace API for notebooks & jobs.(Future Scope)

2) Azure account & resource planning (naming / resource groups)

- Create a dedicated **Resource Group** for data platform (e.g., `rg-data-hydraulic-prod`).
- Use environment prefixes: `dev`, `stg`, `prod` (e.g., `stg-adls-hydraulic`, `prod-databricks-ws`).

3) ADLS Gen2 setup

- Create ADLS Gen2 Storage Account.
- Use **private endpoint** for production to avoid public access.
- Use **Azure AD** for authorization (no shared keys in apps).
- Create a Managed Identity or Service Principal for Databricks / ADF, assign roles (Storage Blob Data Contributor/Reader) scoped to the containers or folders.

4) ADLS folder structure

```
/hydraulic-pumps/
  /raw/                    # landed files raw files
  /bronze/                 # initial parsed parquet/delta (raw cleaned)
  /silver/                 # joined, conformed entities (pump master, inventory snapshot)
  /gold/                   # aggregates, BI-ready tables (product KPIs, stock levels)
  /schemas/                # example JSON schema files
  /logs/                   # ingestion logs
```

5) Databricks workspace setup

- Create workspace in same region as storage.
- use private link for production workspaces.
- Create **Databricks workspace** -> create **clusters** with appropriate node types.
- Configure **secret scope** (backed by Key Vault) with service principal credentials to access ADLS .
- Set up Unity Catalog for table-level governance.

6) Data ingestion pattern (Bronze / Silver / Gold)

- **Bronze**: Raw → minimally parsed → write Delta tables partitioned by ingestion date. Keep original file reference and ingestion metadata (filename, received_ts).
- **Silver**: Clean, typed, deduped, normalized. Join reference data (e.g., pump model lookup), handle slowly changing dimensions (SCD2) for product master.

- **Gold**: Business-ready aggregates and views for BI/Power BI (current inventory, reorder alerts, metrics).

Bronze write example (PySpark → Delta):

7) Orchestration & job schedule

- Use **ADF** if you have many heterogeneous sources; use Databricks jobs if transformations are Databricks-native.
- Recommended pattern: ADF trigger -> copy (if needed) -> start Databricks job for transform -> post-checks -> notify via Teams/email.(Future scope)
- Use job retries, idempotency, and alerting on failures.(future scope)

8) Schema management & data catalog

- Register Delta tables in meta-store.
- Use Azure Purview or Unity Catalog to store dataset metadata, schema, and lineage.

9) Security & networking

- Use **Service Principals** — no storage keys in code.
- Use **private endpoints** for ADLS & Databricks (workspace) in prod.
- Use **Key Vault** for secrets; sync with Databricks Secret Scopes.

10) CI/CD & reproducibility

- Git Repo layout :

```
/dta-etl-repo
  /notebooks         # .py or .dbc notebooks (source)
  /jobs              # job json definitions
  /tests             # unit tests for ETL (pytest future scope)
  /deploy            # scripts to deploy to databricks workspace(future scope)
```

Data Analysis:

- **Stock level summary** (current pump quantities, pump parts quantities, reorder needs, shortages)
- **Valuation** (inventory value by pump type & its parts)
- **Trends** (sales or usage over time, on historical data)
- **Operational efficiency** (turnover rates, aging stock, etc.)
- **Analysis Metrics per Part**

1. **Total Value ($)** = Quantity × Unit Cost → shows the financial impact of each part.
2. **Pump & part Stock Status** → helps identify:
    - Low stock → consider reorder
    - High stock → monitor for slow-moving items
    - Moderate stock → maintain level

Inventory Table: