⛏️ Coal Production ETL Pipeline

📌 1. Project Overview

This project was built as part of a technical challenge from a hiring company. It demonstrates a complete ETL (Extract, Transform, Load) pipeline that processes coal production and equipment sensor data, stores it into a data warehouse (Apache Doris), and visualizes insights via Metabase. The pipeline also includes a forecasting component using Prophet.

⚙️ 2. Tech Stack

- **Python 3.11**
- **MySQL 8.0** (for raw/staging data)
- **Apache Doris** (data warehouse)
- **Metabase** (BI visualization)
- **Docker & Docker Compose**
- **Prophet (forecasting)**

🔄 3. ETL Pipeline Design

📥 Extract

- Extract production and equipment data from MySQL using `extract.py`
- Supports multiple tables and CSV input

🔧 Transform

- Clean null/missing values
- Convert units, format dates
- Aggregate production per day/mine
- Create derived features like:
    - Cumulative volume
    - Production rate
    - Sensor statistics

📤 Load

- Load transformed data into Apache Doris into 2 fact tables:
    - `daily_production_metrics`
    - `production_per_mine`

✅ Validate

- Basic sanity checks before loading
- Logs validation failures to `logs/etl_errors.log`

📁 4. File Structure

```
.
├── main_etl.py                  # Entrypoint for ETL process
├── production_forecast.py       # Forecast future production volume using Prophet
├── config/
│   └── db_config.py             # DB configuration (MySQL & Doris)
├── etl/
│   ├── extract.py               # Extraction logic
│   ├── transform.py             # Transformation logic
│   ├── load.py                  # Load logic to Doris
│   └── validate.py              # Data validation
├── mysql-init/
│   └── production_logs.sql      # Initial raw table (import manually to MySQL)
├── dashboard/
│   ├── daily_production_linechart.png
│   ├── quality_bar_chart.png
│   └── rainfall_scatter.png     # Metabase export samples
├── forecast/
│   ├── forecast_plot.png
│   └── forecast_tomorrow.csv    # Prophet output
├── data/
│   └── equipment_sensors.csv    # Sample CSV for external sensor data
├── logs/
│   └── etl_errors.log           # Error log from validation stage
├── docker-compose.yml           # Run MySQL, Doris, and Metabase containers
├── requirements.txt             # Python dependencies
├── docs/
│   ├── documentation_report.md  # Full documentation
│   └── pipeline_design.md       # Design diagrams / explanations
├── README.md                    # Main project readme
```

▶ 5. How to Run

```
 # Setup virtual environment
python -m venv venv
source venv/bin/activate  # or venv\Scripts\activate on Windows

# Install Python dependencies
pip install -r requirements.txt

# Import production_logs.sql manually to MySQL
mysql -u root -p < mysql-init/production_logs.sql

# Run the ETL
python main_etl.py

# Run forecasting
python production_forecast.py

# Start MySQL, Doris, and Metabase services
docker-compose up -d

# Run the ETL
python main_etl.py

# Run forecasting
python production_forecast.py
```

6. Data Schema

MySQL (Staging)

- `production_data(mine_id, date, volume, unit)`

- `equipment_sensors(mine_id, date, temperature, vibration, moisture)`

Apache Doris (Transformed)

- `daily_production_metrics(date, total_volume)`
- `production_per_mine(mine_id, date, volume)`
- `equipment_summary(mine_id, date, avg_temp, avg_vibration, avg_moisture)`

7. Visualization (Metabase)

Visual dashboards created using Metabase can include:

- Daily production line chart
- Production per mine bar chart
- Scatter plot of sensor readings vs volume
- Future production forecast (Prophet output)

> Metabase is accessible at http://localhost:3000

8. Forecasting Module

The `production_forecast.py` script:

- Uses Prophet to model and predict daily production trends
- Saves:
  - `forecast_plot.png`
  - `forecast_tomorrow.csv`

Limitations:

- Simple univariate time series model
- Can be enhanced with multivariate or deep learning approaches

9. Known Issues & Limitations

- `mysqlclient` or `pymysql` require native system libs; may fail in some Docker builds
- Doris query engine may require tuning for complex aggregations
- Metabase setup is manual; dashboards not created automatically
- Validation is basic (extendable with schema or data drift checks)

10. Conclusion

This project successfully demonstrates:

- Building an end-to-end ETL pipeline
- Loading to a real data warehouse (Doris)
- Forecasting and dashboarding using Python and Metabase

Although some components are simplified or manual (e.g., data import), the architecture and structure are scalable and modular for production-level use.

11. Credits

This project was initially part of a hiring challenge from a real-world company (name undisclosed). It has been extended and repurposed into a personal portfolio project to demonstrate practical data engineering and ETL skills.