AI Engineer Challenge: Coal Mining Operation Analysis Data Pipeline

July 31, 2025

Haikal Ramadhan Usman

1. Background

The purpose of this project is to provide insights regarding production, coal quality, equipment utilization, fuel efficiency and predict next day production. This report documents and explains all process of this project including data pipeline design, ETL workflows, dashboard and prediction result.

2. Scenario

A coal mining company aims to optimize its mining operations using production data. The task is to design and implement a data pipeline that collects, transforms, and loads coal production data from various sources into a data warehouse.

3. Data Pipeline Design

This data pipeline is designed using container-based architecture with Docker Compose to ensure portability, scalability and environment isolation. The diagram design is in Data Pipeline.pdf file. The main components and their flows are follows:

3.1. Pipeline Architecture

a. **Data Sources**
   - **SQL Database**: A table production_logs with columns log_id, date, mine_id, shift, tons_extracted, quality_grade.
   - **IOT Sensors**: A CSV file equipment_sensors.csv with columns timestamp, equipment_id, status, fuel_consumption, maintenance_alert.
   - **Weather API**: Provides daily weather data for Berau, Kalimantan, Indonesia (latitude: 2.0167° N, longitude: 117.3000° E) via the Open-Meteo API endpoint

b. **ETL Engine**
   - **Pyspark**: used as the data processing engine. Pyspark was chosen for its ability to process large-scale data.
   - **Python** scripts (equipment_sensors.py, weather_api.py, calculate_daily_metrics.py) containing the ETL logic.
   - **Mysql** script (1_production_logs.sql, 2_create_equipment_table.sql, 3_create_weather_table.sql) to Create coal_mining DB, mines, production_logs, equipment_sensors, weather_data tables and insert production_logs and mines data.

c. **Data Warehouse**: **MYSQL** is used as a relational data warehouse to store ingested raw data and transformed metrics.

d. **BI Tools: Metabase** is used as a visualization and dashboarding tool to expose insights from the processed data.

e. **Orchestration**
   - **Docker Compose**: Orchestrates all services (MySQL, Metabase, ETL App) in an isolated environment, managing networks, volumes, and container startup.
   - **run_etl_scripts.sh**: A simple shell script running inside the ETL container to sequence the execution of PySpark scripts.

3.2. Data Flow

a. **Initialization**:
   - 1_production_logs.sql creates database coal_mining and create table mines and production_logs and inserts the data to those tables
   - 2_create_equipment_table.sql creates table called equipment_sensors
   - 3_create_weather_table.sql creates table called weather_data

b. **Extract:**
   - equipment_sensors.py reads sensor data from a local CSV file.
   - weather_api.py fetches weather data from an external API.
   - calculate_daily_metrics.py reads raw data from tables in MySQL.

c. **Transform:**
   - **Data Cleaning**: Handling negative values (tons_extracted set to 0), data type conversion (object to float).
   - **Generated Metrics:**
     1. **total_production_daily**: Total tons of coal mined per day.
     2. **average_quality_grade**: Average coal quality per day.
     3. **equipment_utilization_pct**: Percentage of time equipment per day.
     4. **fuel_efficiency**: Average fuel consumption per ton of coal mined.
     5. **daily_rainfall_mm, average_temp_daily, is_rainy_day**: Weather metrics.

d. **Load:**
   - Ingested sensor and weather data are loaded into equipment_sensors and weather_data tables in MySQL.
   - All transformed metrics are loaded into the daily_production_metrics table in MySQL (using overwrite mode to ensure the latest data).

4. ETL Process

a. **Database Initialization**: The MySQL database is initialized with the necessary table schemas via SQL scripts in the synapsis/init folder, executed by Docker during MySQL container startup.

b. **Raw Data Ingestion**:
   - equipment_sensors.py: Reads equipment_sensors.csv and loads it into the equipment_sensors table.
   - weather_api.py: Fetches weather data and loads it into the weather_data table.

c. **Data Transformation** (using calculate_daily_metrics.py):
   - **Data Reading:** Reads production_log, equipment_sensors, and weather_data from MySQL.
   - **Data Cleaning:**
     1. **tons_extracted:** Negative values are replaced with 0.
     2. **All numerical columns:** Converted to float type, and comma thousands separators are removed.
   - **Metric Calculation:**
     1. total_production_daily and average_quality_grade are calculated from production_log.
     2. equipment_utilization_pct is calculated from equipment_sensors by considering the duration of 'active' status.
     3. fuel_efficiency is calculated from equipment_sensors and production_log.
     4. daily_rainfall_mm, average_temp_daily, is_rainy_day are aggregated from weather_data
   - **Data Merging:** All metrics are merged into a single DataFrame based on date.
   - **Writing to Data Warehouse:** The final metrics DataFrame is loaded into the daily_production_metrics table in MySQL.
   - **Data Transformation Validation:**

1. **Negative/Anomaly Values:** Verifying that negative tons_extracted values have been replaced with 0, and equipment_utilization_pct is within the 0-100% range.
2. **Data Types:** Checking df.info() and df.describe() in PySpark to ensure columns have the correct data types (float64 for numerical metrics)

5. Production Data Prediction Model

This script is created to predict the production data for the next production. You can check Next Production Prediction.ipynb file to see the analysis.

6. Deliverables

- Data pipeline design document (in PDF format) is in file synapsis/Deliverables/Data Pipeline.pdf
- Python script and SQL queries (if applicable).
  - synapsis\init\1_production_logs.sql to create coal_mining database and production_logs and mines table and load the data
  - synapsis\init\2_create_equipment_table.sql to create equipment_sensors table
  - synapsis\init\3_create_weather_table.sql to create weather_data table
  - synapsis\etl\weather_api.py extract data from api, transform and load to weather _data table
  - synapsis\etl\ equipment_sensors.py extract data from equipment_sensors.csv and load to equipment_sensors table
  - synapsis\etl\calculate_daily_metrics.py extract
  - synapsis\Deliverables\daily_production.sql query for metabase to show daily production trends (total_production_daily) over one month.
  - synapsis\Deliverables\average.sql query for metabase to show Comparison of average_quality_grade across mines (mine_id).
  - synapsis\Deliverables\correlation.sql query for metabase to show Relationship between rainfall (rainfall_mm) and daily production (total_production_daily)
- Dockerfile and docker config file (if any).
  - synapsis/docker-compose.yml This configuration file defines and orchestrates the entire multi-container data pipeline application, detailing how the MySQL database, the ETL processing application, and the Metabase BI tool are set up, linked, and managed as interconnected services
  - synapsis/etl/ Dockerfile This file serves as the blueprint for building the etl_app Docker image, specifying the base Python environment, Apache Spark installation, and inclusion of all necessary ETL Python scripts and dependencies
- Dashboard (screenshot or link).
  - synapsis\Deliverables\daily_production.png Line Chart Daily production trends (total_production_daily) over one month
  - synapsis\Deliverables\Average.png Bar Chart Comparison of average_quality_grade across mines (mine_id)
  - synapsis\Deliverables\Correlation.png Scatter Plot Relationship between rainfall (rainfall_mm) and daily production (total_production_daily)
- Production data prediction model
  Next Production Prediction.ipynb
  - Documentation report: This document itself