ETL PROJECT REPORT

BOUHOUCHE Ahmed Rami MIMI03

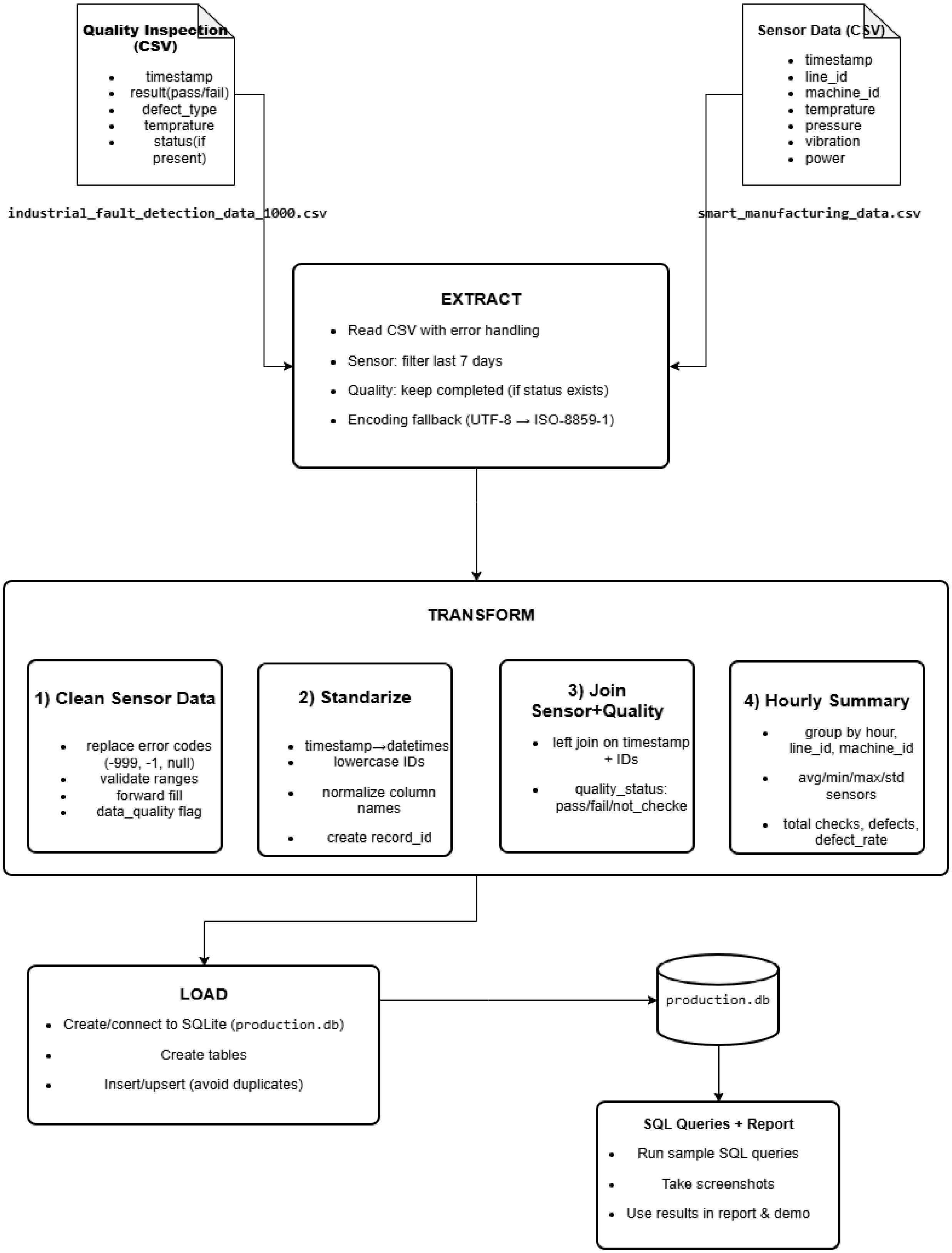
KHORCHI Anis MIMI03

SI HADJ MOHAND ALI MIMI03

GitHub link in case of problems:

https://github.com/a-ramibouhouche/etlpipeline.git

The Extract-Transform-Load Data flow Diagram:



2. Transformation Logic

After extracting the sensor and quality inspection datasets from CSV files, several transformation steps were applied to clean, standardize, and integrate the data before loading it into the database.

2.1 Sensor Data Cleaning

Sensor data often contains invalid or missing values due to temporary sensor failures or communication issues.
To address this:

- Known error codes such as -999, -1, and null values were replaced with missing values.
- Sensor readings were validated against realistic operating ranges:
  - Temperature: 0–150 °C
  - Pressure: 0–10 bar
  - Vibration: 0–100 mm/s
- Invalid values were forward-filled using the previous valid reading. This approach reflects real industrial monitoring scenarios where the last known valid value is often a reasonable estimate.
- A data_quality flag was added to indicate whether a value was directly measured, estimated, or invalid.

2.2 Data Standardization

To ensure consistency across datasets:

- All timestamps were converted to a unified datetime format.
- Column names were normalized (lowercase, no spaces).
- Machine and line identifiers were standardized.
- A unique record_id was generated for each sensor reading to support idempotent loading and avoid duplicate records.

2.3 Join Operation Between Sensor and Quality Data

Sensor readings were combined with quality inspection data using a **left join** based on matching timestamps and machine or line identifiers.

The left join strategy was intentionally chosen to:

- Preserve all sensor readings, even when no quality inspection occurred at the same time.
- Avoid data loss, since quality inspections may be less frequent than sensor measurements.

A new column, quality_status, was introduced to indicate whether a corresponding inspection resulted in a pass, fail, or was not available.

2.4 Hourly Aggregation

To support efficient analysis, hourly summary metrics were computed:

- Data was grouped by hour, line ID, and machine ID.
- Aggregate statistics such as average, minimum, and maximum sensor values were calculated.
- Quality metrics including total inspections, defect counts, and defect rate percentages were derived.

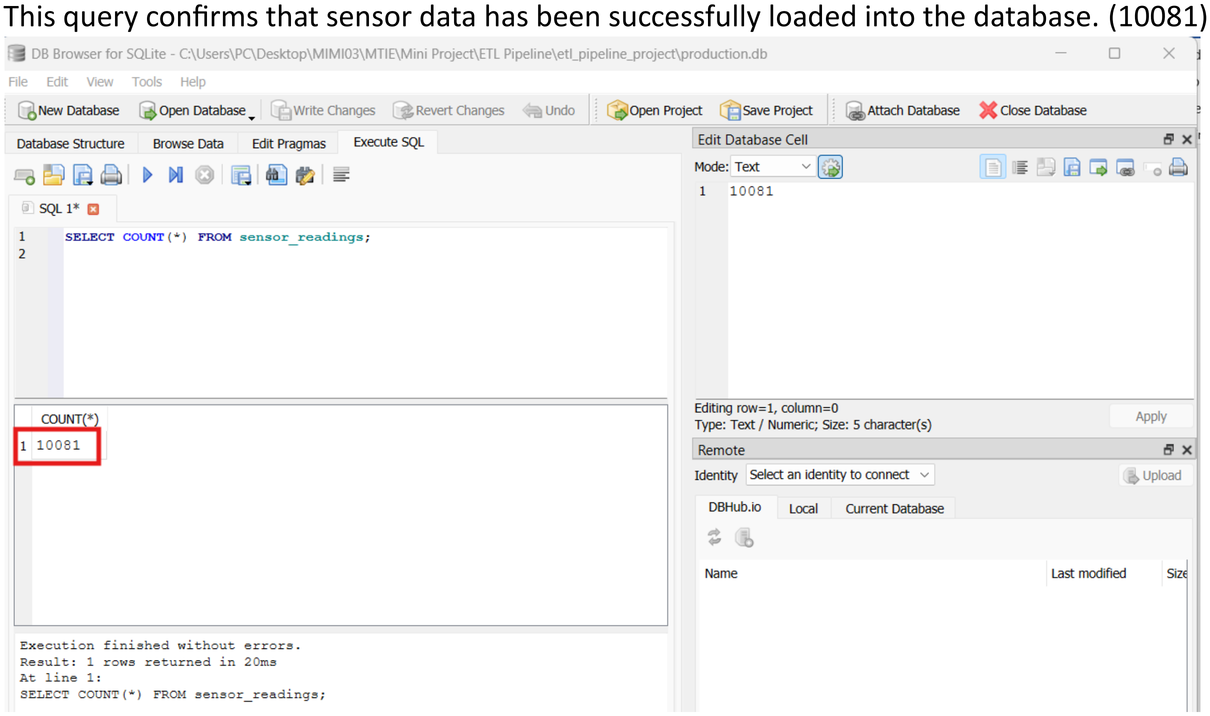
This separation between raw data and aggregated summaries allows both detailed analysis and high-level monitoring.

3. Sample Queries and Results

After loading the transformed data into the SQLite database, several SQL queries were executed to validate the pipeline and demonstrate its analytical capabilities we used DB Browser(SQLite) for that.

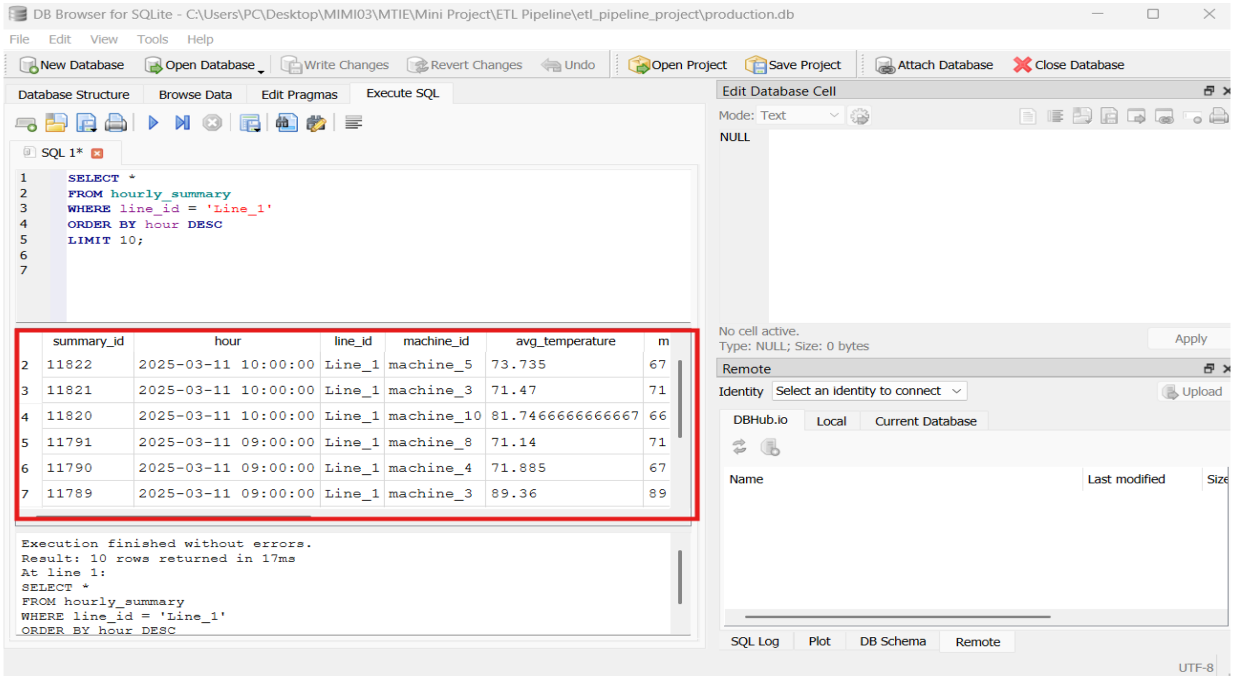
3.1 Total Number of Sensor Records

This query confirms that sensor data has been successfully loaded into the database. (10081)



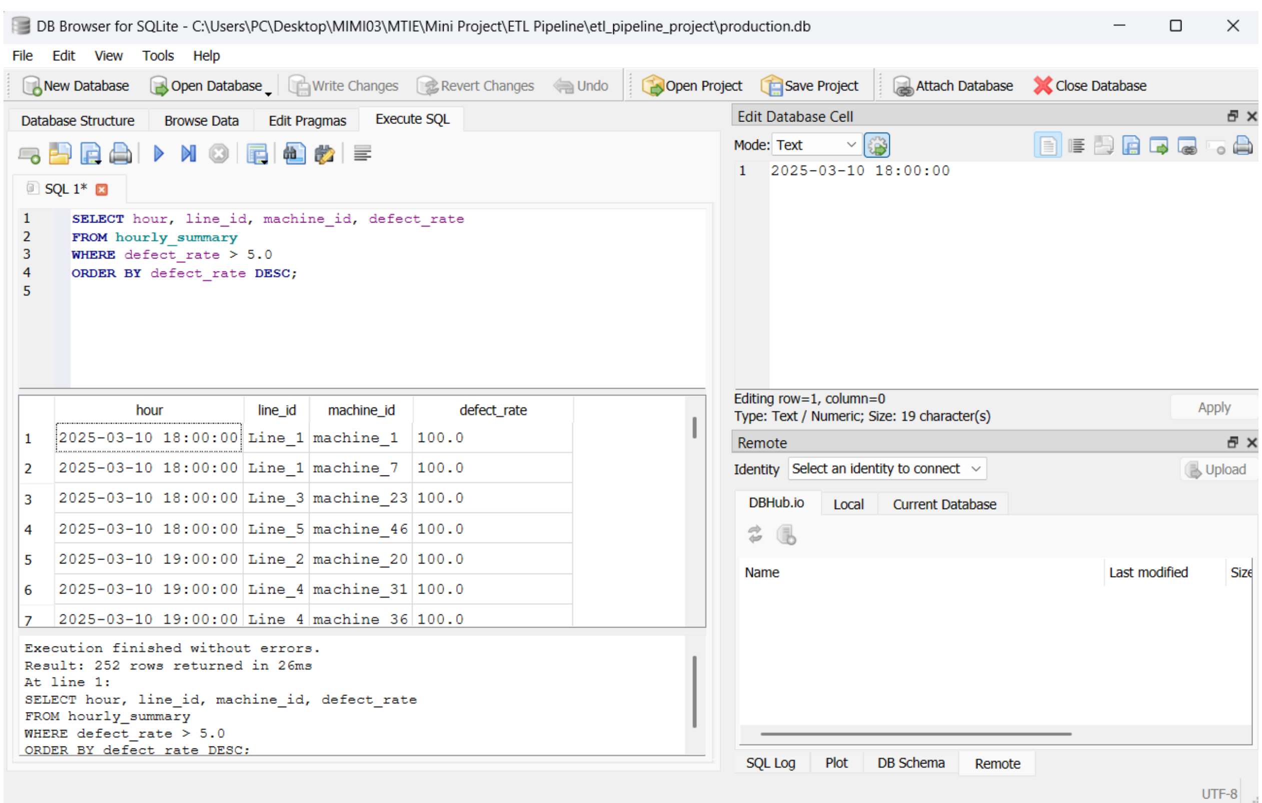
3.2 Hourly Summary for a Specific Production Line

This query retrieves recent hourly aggregated metrics for a specific production line, enabling operational monitoring.



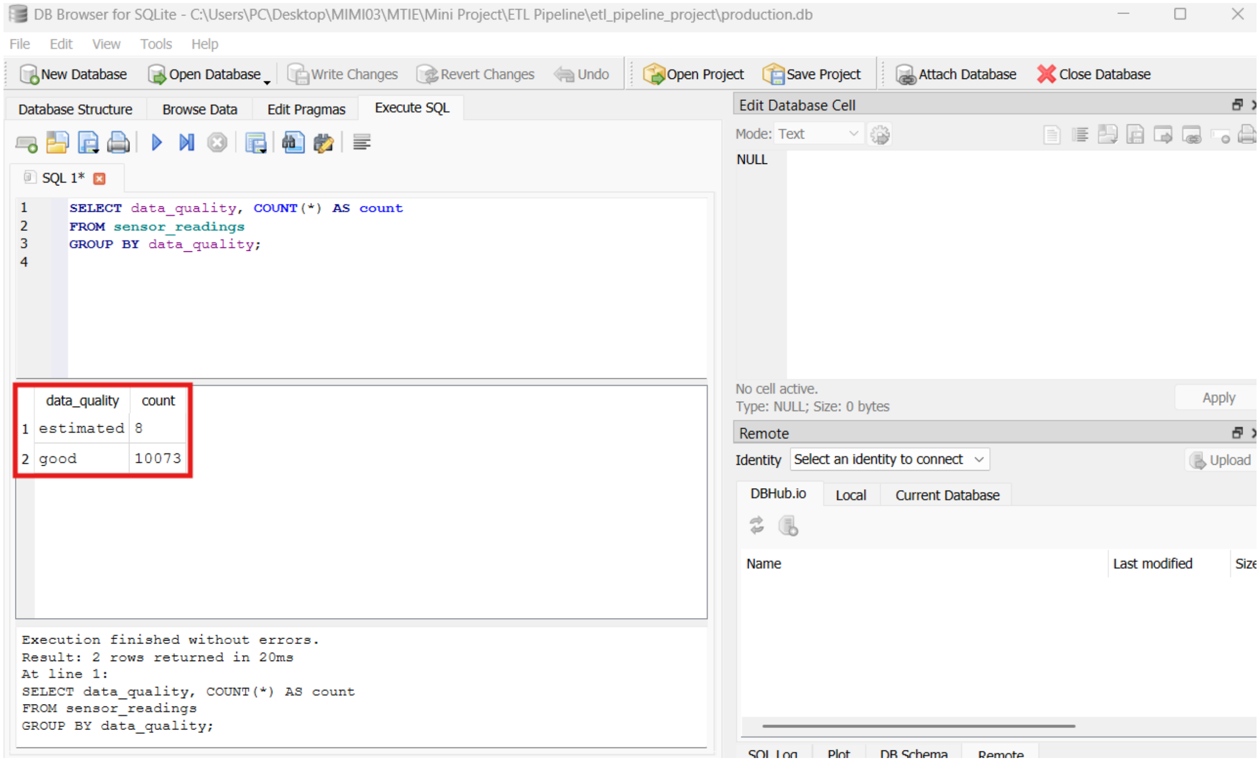
3.3 Identification of High Defect Rate Periods

This query highlights time periods and machines with unusually high defect rates, which can help prioritize maintenance or investigation efforts.



3.4 Data Quality Distribution

This query provides insight into the overall quality of the sensor data after transformation.



4. Challenges and Solutions

4.1 Handling Invalid and Missing Sensor Values

Challenge:
Raw sensor data contained invalid readings and missing values that could distort analysis.

Solution:
Validation rules and forward-filling strategies were applied to clean the data while preserving continuity in sensor measurements. A data quality flag was added to maintain transparency.

4.2 Aligning Sensor and Quality Data

Challenge:
Sensor readings and quality inspections were recorded at different frequencies, leading to incomplete matches.

Solution:
A left join strategy was used to retain all sensor data while attaching quality results only when available. This avoided unnecessary data loss.

4.3 Ensuring Reproducibility and Idempotency

Challenge:
Re-running the ETL pipeline could result in duplicated records.

Solution:
Unique identifiers were generated for sensor records, and database constraints were used to support idempotent loading.

4.4 Environment and Execution Issues

Challenge:
Differences in operating systems and file paths caused execution errors during development.

Solution:
Clear command-line execution steps and dependency management through a virtual environment ensured consistent execution.

Final Conclusion

This project demonstrated the design and implementation of a complete ETL pipeline for integrating production sensor data and quality inspection data into a centralized SQLite database. By extracting data from multiple CSV sources, applying structured transformation and cleaning logic, and loading the results into a normalized database schema, the pipeline enables reliable monitoring and analysis of manufacturing operations.

Special attention was given to data quality, reproducibility, and scalability. Invalid sensor readings were handled using validation rules and estimation strategies, while a left join approach ensured that all operational data was preserved even when quality inspections were unavailable. The separation between detailed sensor data and aggregated hourly summaries allows both low-level inspection and high-level performance analysis.

Overall, this project reflects a practical, production-oriented approach to data engineering. The resulting system is reproducible, extensible, and capable of supporting operational decision-making through structured SQL queries and reporting.