Distributed Sales Data Processing System – Summary

# Overview

This project implements a distributed system for processing and analyzing a very large sales dataset (5 million rows). The system follows a server–worker architecture:

- The server (coordinator) splits the dataset into chunks and distributes them.

- Multiple worker nodes compute partial results on assigned chunks.

- The server aggregates results from workers and produces the final statistics.

- All intermediate and final results are stored in an SQL database (SQLite in this implementation).

# Dataset

- Sample dataset: 5 million transactions (CSV format).

- Each row contains at least:

• Price

• Quantity

- The main metric of interest is Sales Amount = Price × Quantity.

# System Components

The system is divided into three main components: Database, Server, and Worker Nodes.

# 1. Database (SQLite)

Two tables are used:

- worker\_results:

• Worker ID

• Rows processed

• Total sales amount

• Min, Max, Avg Price

• Timestamp

- final\_results:

• Total rows processed

• Total sales amount

• Global Min, Max, Avg Price

• Processing time

• Workers used

• Chunks processed

• Timestamp

This ensures both partial and final statistics are stored and can be queried later.

# 2. Server (Coordinator)

Responsibilities:

1. Loads the full dataset with Pandas.

2. Splits it into equal-sized chunks (10 per worker).

3. Waits for all workers to connect via TCP sockets.

4. Sends each worker their assigned chunks.

5. Receives partial statistics from workers.

6. Inserts partial results into the database.

7. After all workers finish, aggregates final results and prints a formatted summary.

Key features:

- Uses pickle for serialization.

- Uses struct to send message lengths safely.

- Implements logging for transparency.

# 3. Worker Nodes

Responsibilities:

1. Connect to the server over TCP.

2. Wait for assigned DataFrame chunks.

3. For each chunk:

- Compute rows processed, total sales amount, min, max, avg price.

- Send results back to the server.

4. Exit when a STOP signal is received.

# Parallelism & Communication

- Supports multiple worker nodes (2+).

- Server listens for connections and distributes work.

- Workers run in parallel threads and process chunks concurrently.

- Uses send\_msg / recv\_msg helpers to handle reliable data transmission.

# Example Run

- Dataset: 5,000,000 rows

- Workers: 3

- Chunks: 10 per worker (30 total)

Final Aggregated Results:

- Total Rows: 5,000,000

- Total Sales Amount: ~827M

- Min Price: 0.15

- Max Price: 3226.11

- Avg Price: 33.10

- Processing Time: ~31 seconds

# Key Features Implemented

- Automatic dataset loading (sales\_data\_5m.csv).

- Equal chunk distribution across all workers.

- Partial results stored in SQL database for reproducibility.

- Final aggregation done using SQL queries for accuracy.

- Concurrency & sockets for distributed processing.

- Error handling for dropped or failed worker connections.

- Formatted output report for easy interpretation.

# Conclusion

This distributed system demonstrates:

- Efficient handling of large datasets using parallelism.

- Proper use of client-server architecture with sockets.

- Reliable serialization of Pandas DataFrames with pickle.

- Persistent storage of both partial and final results in SQL.

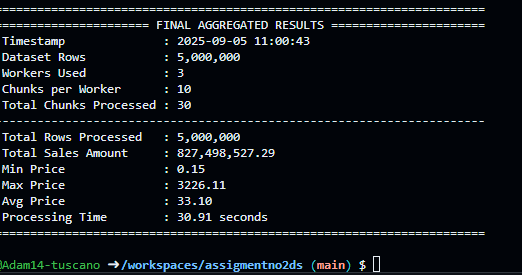
- Scalability to handle more workers and larger datasets.

It meets the assignment requirements for parallelism, socket communication, SQL integration, and final aggregated output.

**Server Output**

The following screenshots show the server execution and final aggregated results.

# 



**Worker Outputs**

The following screenshots show the outputs from three different workers.

