Advanced Trade Reconciliation & Data Processing System

Objective:

Build a **Python-based trade reconciliation system** that:

1. **Ingests client orders from an SQLite database (`trades.db`)**
2. **Processes trade files from brokers (in `.eml` format with Excel attachments)**
3. **Splits quantity, brokerage, STT, as reported by the broker files to clients as per the trades db.**
4. **Handles complex reconciliation cases** like:
    - Multiple brokers executing trades for the same stock
    - Partially matched and incomplete trades
    - Trades exceeding or below client orders
5. **Stores data in a database** and generates reports
6. **Schedules the script to run automatically**

Scope of the Assignment

1. Trade Data Sources

Client Orders (Stored in `trades.db`)

- This SQLite database contains client orders for various stocks.
- Candidate must **extract the data** and load it into a Pandas DataFrame.
- Expected structure:

order_id	client_id	symbol	quantity	order_price	order_date
ORD001	C001	XYZ	100	101.5	2025-01-01
ORD002	C002	ABC	200	50.25	2025-01-01
ORD003	C003	XYZ	50	102.0	2025-01-02

Broker Trade Files (Stored in `.eml` files)

- Each broker **emails** trade execution files daily.
- These emails contain **Excel attachments** with the executed trade details.
- Candidate must **extract and parse** these files.

Each trade file contains:

Deal Date	Party Code/SEBI Regn Code	Instrument ISIN	Buy/Sell Flag	Quantity	Cost	Col 8	Net Amount	Brokerage Amount	Settlement Date	STT	Exchange Code	Depository Code
12/06/24	BROKER1CODE	INE700A01033	B	12000	1102.52		13240152.96	3175.362	13/06/24	6728	NSE	NSDL
12/06/24	BROKER1CODE	INE738I01010	B	680	3596.65		2451306.43	1791.0385	13/06/24	3794	NSE	NSDL
12/06/24	BROKER1CODE	INE457L01029	B	5358	980.094		5253027.46	539.804	13/06/24	1144	NSE	NSDL
12/06/24	BROKER1CODE	INE665J01013	B	581	3878.58		2269389.33	5109.3474	13/06/24	10825	NSE	NSDL
12/06/24	BROKER1CODE	INE982J01020	S	30000	984.88		29541464.00	1557.016	13/06/24	3298	NSE	NSDL
12/06/24	BROKER1CODE	INE00F201020	B	1740	2945.03		5137607.20	4250.744	13/06/24	9006	NSE	NSDL
12/06/24	BROKER1CODE	INE0NT901020	B	7308	2708.54		19819077.02	8033.5456	13/06/24	17020	NSE	NSDL
12/06/24	BROKER1CODE	INE321T01012	B	900	2569.16		2320189.29	2549.0654	13/06/24	5400	NSE	NSDL

---

2. Tasks to Perform

A. Extract and Parse Data

1. **Load Client Orders**
    - Read `trades.db` using SQLite and convert it into a Pandas DataFrame.
2. **Extract Trade Files from Emails**
    - Read `.eml` files and extract attached `.xlsx` files.
    - Load these trade files into Pandas.

B. Trade Reconciliation

Match broker trades with client orders based on:

- **Symbol**
- **Date**
- **Quantity Matching Logic:**
    - If **exact match** → Mark as matched.
    - If **partial match** → Split and allocate.
    - If **broker quantity exceeds order** → Flag as excess.
    - If **order not filled** → Mark as pending.

Edge Cases:

- **Multiple brokers** for the same stock → Allocate proportionally.
- **Trades executed at different prices** → Calculate **average price** for reconciliation.
- **Different brokerage rates** → Apply correct cost structure.

C. Cost Calculation

For each trade:

- **Brokerage Cost**: Use the "Brokerage Amount" column from the trade file for each transaction.
- **STT (Securities Transaction Tax)**: Use the "STT" column directly from the trade file.
- **Total Cost** = `Net Amount + Brokerage Amount + STT` (all values provided directly in the trade file).
- **Execution Slippage** = `order_price - (Net Amount / Quantity)`.

D. Database Integration

Store the following in SQLite:

- **Client Orders (`client_orders` table)**
- **Broker Trades (`broker_trades` table)**
- **Reconciliation Results (`reconciliation_results` table)**

E. Reporting

Generate **daily reconciliation reports**:

1. **Matched Trades (`matched_trades.csv`)**
2. **Unmatched Trades (`unmatched_trades.csv`)**
3. **Broker Comparison (`broker_summary.csv`)**

F. Automation & Scheduling

1. **Automate Execution**:
    - Create a script (`run_reconciliation.py`) that:
        - Extracts data
        - Reconciles trades
        - Updates database
        - Generates reports

2. **Schedule Execution**:
    - **Linux:** Use `cron` (`0 18 * * * python3 run_reconciliation.py`)
    - **Windows:** Use Task Scheduler

Deliverables:

1. **Python Scripts**
    - `extract_trades.py`: Extracts client orders and broker trades.
    - `reconcile_trades.py`: Matches trades and calculates costs.
    - `generate_reports.py`: Generates CSV summaries.
    - `run_reconciliation.py`: Automates the full workflow.
2. **Database (`trades.db`)
    - Updated with reconciliation results.
3. **Reports**
    - `matched_trades.csv`
    - `unmatched_trades.csv`
    - `broker_summary.csv`
4. **README**
    - Instructions on running the project.
    - Explanation of reconciliation logic.
    - Scheduler setup.

Evaluation Criteria:

1. **Data Processing Skills**
2. **Trade Reconciliation Accuracy**
3. **Database Management**
4. **Automation & Scheduling**
5. **Code Quality**

Bonus Challenges

1. **Email Integration** - Fetch `.eml` files from an email inbox using `IMAP`.
2. **Live Execution** - Use a Flask API to trigger reconciliation on demand.
3. **Broker Ranking** - Add a ranking system for brokers based on execution quality.