COMP1314 Data Management Coursework: Gold Price Tracker (Instruction Manual)

Tutor:

Dr Fairuz Safwan Mahad

F.S.Mahad@soton.ac.uk

University of Southampton Malaysia

Faculty of Electronics and Computer Science

Prepared by:

Ong Hui Min

hmo1e25@soton.ac.uk

University of Southampton Malaysia

BCs Computer Science Part 1

Faculty of Electronics and Computer Science

15 December 2025

TABLE OF CONTENTS

1.0 SYSTEM OVERVIEW 3

2.0 DATABASE DESIGN 5

2.1 Table Name: currencies 5

2.2 Table Name: gold_prices 5

2.3 Table Name: unit_prices 6

2.4 Table Name: logs 6

3.0 SYSTEM COMPONENTS 8

4.0 SCRIPTING 9

4.1 Script 1: Database Schema Setup 9

4.2 Script 2: Gold Price Tracker (Scraping & Processing) 9

4.3 Script 3: Database Insertion (MySQL) 10

4.4 Script 4: Data Visualisation 11

5.0 LOGGING MECHANISM 12

6.0 AUTOMATION WITH CRONTAB 13

7.0 CONCLUSION 14

1.0 SYSTEM OVERVIEW

The Gold Price Tracker System is a Bash-based data collection and visualisation system designed to automatically scrape live gold price data from online sources, store the collected data in a structured relational MySQL database, and generate historical price plots for analysis. The system eliminates the need for manual data collection by providing a fully automated pipeline that handles data retrieval, processing, storage, logging and visualisation.

The system is implemented using a combination of Bash Scripting, MySQL database management and Gnuplot visualisation. Each component plays a specific role within the workflow, ensuring that data is collected accurately, stored consistently, and presented clearly.

The Gold Price Tracker System integrates the following key features:

1. **Web data scraping**

Live gold price data is retrieved from Kitco, a widely used financial information source, ensuring realistic and up-to-date market data.

2. **Currency conversion**

Gold prices are converted from USD into multiple supported currencies using stored exchange rates.

3. **Structured database storage (MySQL)**

All data is stored in a normalised relational database that follows a clearly defined Entity Relationship Diagram (ERD).

4. **Automated logging**

System activities and insertion events are recorded to support debugging, validation, and auditing.

5. **Data visualisation using Gnuplot**

Historical gold price trends are presented as graphical plots to support analysis and comparison.

6. **Optional scheduling using crontab**

The system can be configured to run automatically at fixed intervals, such as hourly or daily.

Ong Hui Min hmo1e25

The system architecture strictly follows the relational design shown in the Entity Relationship Diagram (ERD), ensuring proper normalisation and consistency between scripts and database structure.

2.0 DATABASE DESIGN

The database is designed according to relational database principles. It consists of four normalised tables, each responsible for managing a specific type of data within the system. This separation of responsibilities reduces redundancy, improves data integrity, and simplifies querying.

2.1 Table Name: currencies

The "currencies" table stores all supported currencies (AUD, CNY, EUR, and GBP) used by the system along with their exchange rates relative to USD. This table allows gold prices to be converted into different currencies in a structured manner.

Key Fields:

- **currencies_id (Primary Key):** A unique identifier for each currency record.
- **currency_name:** The currency code (e.g. USD, EUR, GBP, AUD, CNY).
- **currency_exchange:** The exchange rate used to convert gold prices from USD to the target currency.

Relationship:

- One currency can be associated with many gold price records.
- This forms a one-to-many (1:M) relationship between currencies and gold_prices.

2.2 Table Name: gold_prices

The gold_prices table stores individual gold price snapshots for each currency at specific timestamps. Each record represents one data collection.

Key Fields:

- **gold_prices_id (Primary Key):** Unique identifier for each gold price record.
- **currency_id (Foreign Key → currencies):** Links the gold price record to its corresponding currency.
- **price_timestamp_ny:** The market timestamp based on New York trading time.
- **script_timestamp_my:** The timestamp representing when the script was executed in Malaysia.

- **bid_price, ask_price, high_price, low_price:** Market price values collected from the source.

Relationship:

- Each gold price record belongs to one currency.
- Each gold price record can be linked to one set of unit prices.

2.3 Table Name: unit_prices

The unit_prices table stores gold prices converted into multiple measurement units. This allows users to view gold prices in different units without recalculating values during analysis.

Key Fields:

- **unit_prices_id (Primary Key)**
- **gold_price_id (Foreign Key → gold_prices)**
- **unit_ounce, unit_gram, unit_kilo**
- **unit_pennyweight, unit_tola, unit_tael**

Relationship:

- Each unit price record corresponds to exactly one gold price record.
- This forms a one-to-one (1:1) relationship with the gold_prices table.

2.4 Table Name: logs

The logs table records system activity, including successful insertions, warnings and errors. This table plays a critical role in debugging, auditing and validating the system.

Key Fields:

- **log_id (Primary Key)**
- **currency_id (Foreign Key, nullable)**
- **gold_price_id (Foreign Key, nullable)**
- **message:** A descriptive log message indicating system actions or errors.
- **timestamp:** Automatically generated log timestamp.

Ong Hui Min hmo1e25

Relationship:

- Log entries may reference both currency and gold price records.
- Foreign keys are nullable to ensure logs are preserved even if related data is deleted.

3.0 SYSTEM COMPONENTS

The Gold Price Tracker System is divided into several scripts, with each script responsible for a specific task within the overall workflow. This design prevents the system from becoming a single script and improves code clarity and organisation.

By separating the system into individual components, maintenance and updates can be performed more efficiently. If an error occurs or a feature needs to be enhanced, changes can be made to the relevant script without affecting the rest of the system.

In addition, this structure allows each component to be tested and debugged independently. This simplifies validation, improves reliability and ensures that each part of the system functions correctly before being integrated into the complete workflow.

4.0 SCRIPTING

This section describes the scripting components used in the Gold Price Tracker System. Each script performs a specific function within the overall overflow, ranging from database initialisation to data scraping, storage, and visualisation. These scripts together form an automated pipeline that ensures reliable data collection, processing, and output generation.

The scripting approach follows a modular design, where responsibilities are clearly separated across multiple Bash scripts. This improves readability, simplifies debugging, and allows individual scripts to be executed and tested independently without affecting the rest of the system.

4.1 Script 1: Database Schema Setup

This script creates and initialises the entire database structure according to the ERD. It defines all tables, primary keys, foreign keys and constraints required for the system.

File using: goldtracker_schema.sql

Instructions:

1. Open the MySQL command-line interface using **mysql -u root**
2. Execute the schema file using **SOURCE goldtracker_schema.sql;**

Expected Output:

- Database goldtracker is created
- All four tables are created successfully
- Foreign key relationships are enforced

4.2 Script 2: Gold Price Tracker (Scraping & Processing)

This script is the core of the system. It retrieves live gold price data, processes it and forwards the processed data to the database insertion script.

File using: gold_tracker.sh

Execution Steps in WSL terminal:

1. cd Script
2. chmod +x gold_tracker.sh
3. ./gold_tracker.sh

Key Operations:

- Checks for active internet connectivity
- Downloads HTML content from Kitco
- Extracts bid, ask, high, and low prices
- Converts prices into multiple units
- Converts prices into supported currencies
- Assigns timestamps for New York market time and Malaysia execution time
- Logs execution details
- Forwards data to the database insertion script

4.3 Script 3: Database Insertion (MySQL)

This script handles all database insertions and ensures that data follows the ERD-defined structure.

File using: gold_scrapper.sh

Insertion Flow:

- Insert or update currency data
- Insert gold price records
- Insert unit price records
- Insert log entries

Safety Controls:

- Prevents insertion of zero or invalid prices
- Ensures timestamps are present
- Maintains referential integrity
- Logs both success and failure events

4.4 Script 4: Data Visualisation

This script generates historical gold price plots using Gnuplot and data retrieved directly from the MySQL database.

File using: plot_gold.sh

Execution Steps in WSL terminal:

1. chmod +x plot_gold.sh
2. ./plot_gold.sh

Features:

- Executes SQL queries dynamically
- Uses Malaysia time formatting
- Generates multiple plots automatically
- Stores all output images in the /plots directory

5.0 LOGGING MECHANISM

The Gold Price Tracker System implements a dual logging mechanism to ensure system transparency, traceability and ease of debugging. Logging is an essential part of the system as it records both execution activities and database operations, allowing errors and system behaviour to be reviewed at a later stage.

Database-level logging is implemented through a dedicated “logs” table, which stores structured information such as execution timestamps, operation status and relevant message. This provides a permanent record of system activity and allows historical analysis of script executions directly from the database.

In addition to database logging, file-based logs are generated during script execution. These logs capture real-time execution details, including successful operations and error messages. The combination of database and file-based logs ensures that issues can be identified quickly, whether they occur during data processing, database insertion or automated execution.

6.0 AUTOMATION WITH CRONTAB

The Gold Price Tracker System supports automated execution using crontab, a time-based job scheduler available in Linux and WSL environments. By using crontab, the gold price tracking process can be executed automatically at fixed time intervals without requiring manual intervention. This allows the system to continuously collect, store and update gold price data, making it suitable for long-term monitoring and historical analysis.

For example, the following crontab entry runs the gold_tracker.sh script once every hour, ensuring that the latest gold price data is regularly captured and stored in the database:

```
0 * * * * /path/to/gold_tracker.sh
```

Note:

For crontab jobs to execute correctly, the machine must remain powered on and must not be in sleep or hibernation mode. If the system enters sleep mode, scheduled tasks will not run until the machine is active again. This requirement is especially important when using crontab within a WSL environment.

7.0 CONCLUSION

The Gold Price Tracker System demonstrates the effective integration of Bash scripting, relational database design and data visualisation to form a complete and automated solution. The system can retrieve live gold price data, process it in a structured manner, and store it reliably in a MySQL database. By following a clear workflow from data collection to storage, the system reduces manual effort and ensures that all data is handled consistently and accurately.

This project also showcases real-world computing concepts such as web data scraping, automated data pipelines, and structured logging. Logging is implemented at both the file and database levels, allowing system activities to be tracked and errors to be identified more easily. In addition, historical gold price data is visualised using Gnuplot, enabling users to observe trends over time and compare prices across different currencies in a clear and meaningful way.

The overall implementation strictly follows the ERD design and maintains proper relationships between all database tables. All coursework requirements related to scripting complexity, database integration, automation, and visualisation are fully fulfilled. More importantly, the project provides practical experience in building a real-world automated system, reinforcing key concepts taught throughout the coursework.