Documentation for Step 1 & Step 2: Binance Data Retrieval and Processing

1. Introduction

- **Context**

This project builds an end-to-end analytics solution on top of Binance Spot market data, focusing on cryptocurrency trading pairs such as BTCUSDT, their current spot prices, and the rules under which they are traded. It is carried out within an Analytics Engineer training program and is structured along a realistic data lifecycle, starting with API-based data collection, continuing with relational and dimensional modeling and ELT pipeline construction, and ending with a Power BI dashboard aimed at non-technical stakeholders.

- **Objectives**

Understanding and documenting the data

The first objective is to perform a structured exploration of the Binance Spot REST API and to document the data made available by the relevant endpoints. This includes retrieving a full snapshot of all spot trading pairs at a given point in time, where each row contains the trading pair code (for example BTCUSDT), the current price of that pair, and the timestamp when this price was captured. In addition, the project analyses the exchange-information endpoint to extract schema and business metadata. These findings are summarised in a technical yet readable report that defines the data sources, their fields and constraints, and explicitly states that this project focuses on spot prices and symbol-level metadata, while more complex feeds such as order book depth or individual trade events are not included.

Designing and implementing the data model

The second objective is to convert the semi-structured JSON responses into a well-defined relational schema in third normal form and to derive from it an analytical star-shaped schema optimised for reporting and dashboarding. In the normalised layer, the model separates core entities into dedicated tables. On top of this normalised design, an analytical schema is created in which a central table collects all price observations together with references to the market and time tables, and where surrounding tables provide descriptive attributes about assets, markets and time periods. Deliverables for this objective include a diagram of the entities and their relationships and scripts to create the tables in a relational database.

Building an automated data pipeline (ELT/ETL)

The third objective is to establish an automated process that regularly retrieves, loads and transforms the data. Programmatic calls are used to fetch both current prices and market descriptions from the Binance interfaces, and the raw responses are first stored in simple holding tables. From there, transformation steps convert values, split nested structures into

columns, connect records and populate both the relational and analytical tables. The result is a repeatable pipeline that can refresh the data model with minimal manual effort.

Creating a PowerBI dashboard and presenting the results

The fourth objective is to make the prepared data and model accessible to non-technical stakeholders through a Power BI dashboard and a short presentation. The dashboard uses the analytical schema to display views such as the highest-priced trading pairs in a chosen quote currency, the overall distribution of prices across all markets and simple comparisons by asset category or trading status. In the final presentation, the project briefly walks through the data sources, the structure of the data model and the pipeline, and highlights key insights that can be drawn from the dashboard, such as the strong role of certain assets and the presence of many low-priced markets.

- **Scope**

The scope of this project covers Binance Spot market data, focusing on current prices for all spot trading pairs at the time of retrieval and the key descriptive information that explains these markets and the assets involved. It includes the design and implementation of a relational and analytical data model, the construction of an ELT process that moves data from the Binance API into this model, and the development of a Power BI dashboard that uses the modeled data to answer basic business questions about the price landscape of the Spot market.

The project does not include more granular or advanced data such as full order book information, detailed trade-by-trade history, or data from derivatives and futures products. It also excludes real-time streaming solutions, predictive or machine-learning models, and production-grade orchestration and monitoring. These topics are recognised as potential extensions but lie outside the scope of the current work in order to keep the project focused and achievable within the given timeframe.

---

2. Methodology & Processing Steps

This document outlines the methodology used to collect, validate, and process cryptocurrency market data using the Binance REST API (https://www.binance.com/fr/binance-api). The objective was to create a robust pipeline that ensures data quality while maintaining high efficiency for specific market queries.

Step 1: Status Quo (Exploration & Retrieval)

Phase 1: Exploratory Data Analysis (Bulk Collection)

In the initial stage, a bulk collection approach was chosen in order to explore the data.

- **Data Volume:** A total of 3,497 unique trading pairs were successfully retrieved in a single API call (memory usage: 54.8 KB)
- **Data Info:**
    - Variable Symbol: The unique identifier for the market pair e.g., BTCUSDT (string)
    - Variable Price: The current market price (string -converted to float later for better processing)
- **Integrity Check:**
    - The analysis confirmed 0 missing values,
    - 0 duplicates
    - 3,497 unique values, confirming that each market pair is represented only once.

Phase 2: Implementation of a Generic Retrieval Function

The goal was to create a generic data retrieval function in order to have data from any market.

- **Purpose:** To allow targeted requests for specific markets (e.g., BTC-USDT) without downloading the entire dataset.
- **Efficiency:** This reduces network latency and memory usage significantly.
- **Flexibility:** The function is designed to handle different endpoints and parameters dynamically, making the code reusable for future analysis requirements.

---

3. Accessible Data (Data Schema)

The accessible data is provided in the generated `binance_extracted_data.csv`. This file represents a cleaned and timestamped version of the API response.

binance_extracted_data

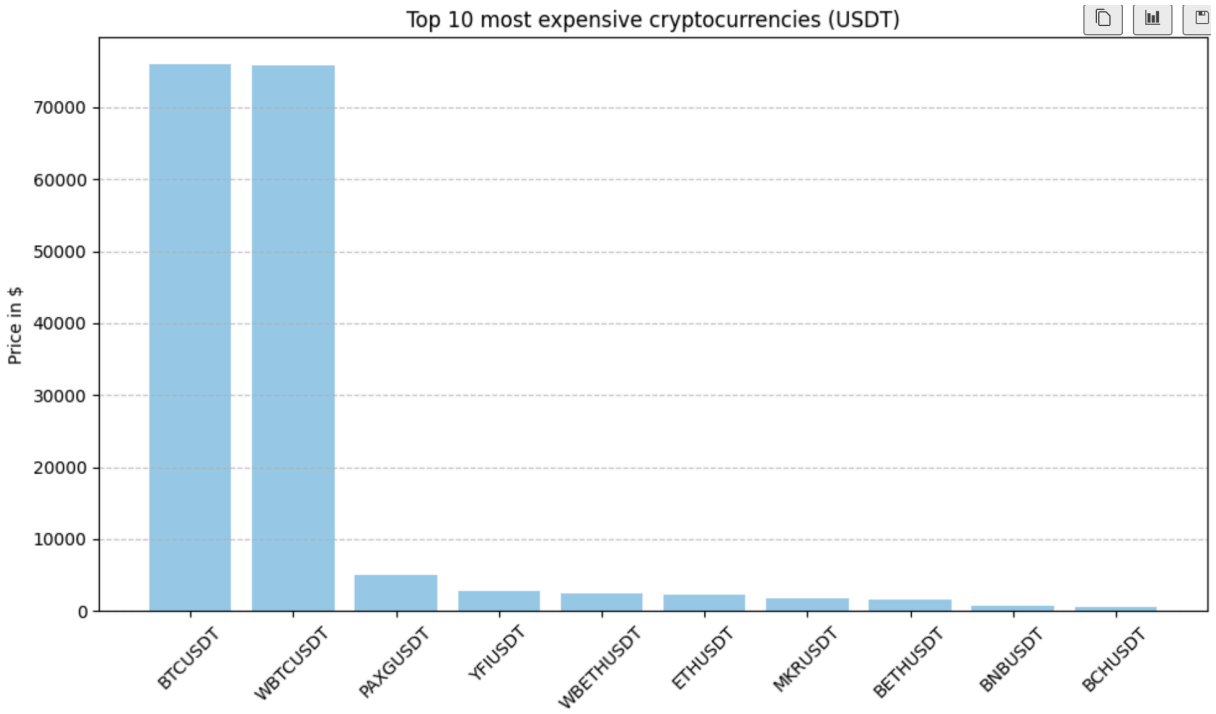
Symbol	Price	Retrieval_Timestamp
BTCUSDT	75799.99000000	2026-02-04 14:24:43
ETHUSDT	2230.42000000	2026-02-04 14:24:43
ETHBTC	0.02943000	2026-02-04 14:24:44
BNBUSDT	748.49000000	2026-02-04 14:24:44
SOLUSDT	95.70000000	2026-02-04 14:24:44

The Price was converted into Float, as mentioned before. Additionally we added the retrieval timestamp when the data was fetched.

---

4. Data Visualization & Insights

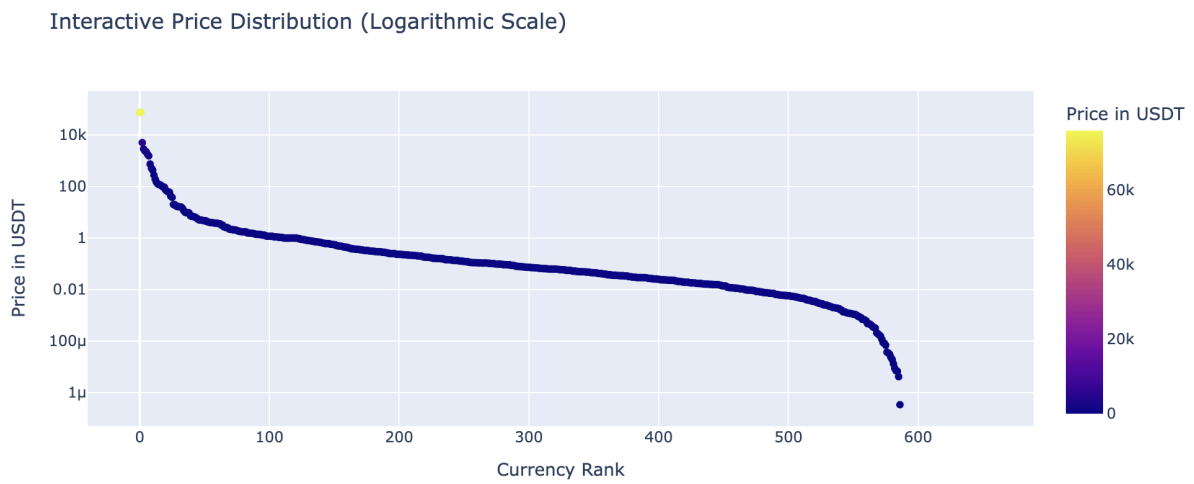
Top 10 Highest Priced Assets (only USD Pairs):



Key Insights

- The "Bitcoin Standard":
    - BTCUSDT (Bitcoin) and WBTCUSDT (Wrapped Bitcoin) lead the market, trading above $75,000.
    - The nearly identical height of these bars confirms that WBTC successfully maintains its 1:1 peg to Bitcoin's price.
- The Price Gap:
    - There is a significant price drop-off after the top two assets. The third-ranked asset (PAXGUSDT) trades below $10,000, highlighting the dominant unit price of Bitcoin.
- Market Diversity:
    - The list includes various types of assets: standard coins (BTC, ETH), wrapped tokens (WBTC, WBETH), and gold-pegged tokens (PAXG), showing the diversity of the 3,497 pairs retrieved.

Price Distribution (logarithmic scale):



Key Insights:

- The Extreme Outlier: Bitcoin (BTC) is isolated at the very top (~$79,000+), dominating the market price structure.
- The Power Law: There is a steep drop-off within the top ranks. Only a handful of assets trade above $1,000.
- The Average: The linear descent in the middle section indicates that the vast majority of established altcoins trade between $0.10 and $10.00.
- The Micro-Values: The tail of the distribution (bottom right) contains assets priced significantly below $0.0001 (e.g., Meme coins with high circulating supply).

---

5. Technical Conclusion (Step 1)

The implementation successfully established a robust data pipeline capable of interacting with the Binance REST API. By validating 3,497 trading pairs with zero missing values, the project demonstrated high data integrity.

The two-phased approach proved to be the optimal strategy:

1. Bulk Retrieval allowed for a comprehensive market audit and the visualization of global price distributions, identifying key market structures like the "Bitcoin Standard" and the long-tail of micro-cap assets.
2. The transition to a Generic Function ensures that future data requests are resource-efficient, scalable, and adaptable to specific analytical needs without redundant network load.

In summary, the system is now fully operational for both high-level market scanning and low-latency, targeted asset monitoring.

Step 2: Data Modelling (Relational System - 3NF)

Approach

In Step 2, we implemented a relational database in **MySQL** to store the data retrieved from the Binance API.

The schema was designed in **Third Normal Form (3NF)** in order to ensure a clean, redundancy-free data foundation and enforce referential integrity through primary and foreign key constraints.

Following this principle, the data was divided into three logical tables:

- **Table: Currency** (Asset_Type: currencies and cryptocurrencies) - Stores currencies (e.g. ID 1 = BTC, ID 2 = USDT).
- **Table: Pairs** - Links currencies into trading pairs (e.g. Pair ID 101 connects Cryptocurrency ID 1 with Currency ID 2).
- **Table: Price_History** - Stores only: Pair_ID, Price and a generated timestamp

Data Enrichment Research

To extend the analytical potential of the Binance API data, additional external data sources were analysed. CoinMarketCap was identified as a relevant enrichment source, as it provides sector-based categorisation of cryptocurrencies and trading pairs.

Such categorisation enables extended analytical use cases, for example evaluating how specific sectors (e.g. DeFi, Stablecoins, Layer 1, Gaming) develop over time or comparing price dynamics across categories.

Based on this research, the following major categories were identified as most relevant

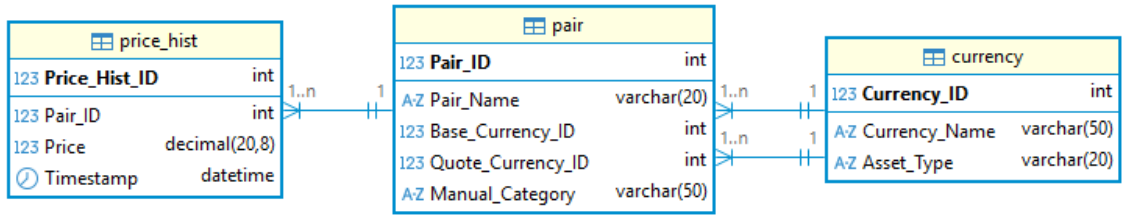
Category	Description	Example (Token/Projekt)	Example trading pairs (Pairs)
DeFi (Decentralized Finance)	Decentralised financial applications without intermediaries (e.g. lending, exchanges, insurance).	Uniswap (UNI)	UNIUSDT, UNIBTC, ETHUSDT
Stablecoins	Cryptocurrencies pegged to stable assets to minimise volatility.	Tether (USDT)	BTCUSDT, ETHUSDT, BNBUSDT, SOLUSDT
NFTs (Non-Fungible Tokens)	Unique digital assets for art, collectibles, and digital goods..	CryptoPunks	ETHUSDT, ETHBTC
Gaming & Metaverse	Tokens for digital games and virtual worlds focused on ownership and digital economies.	Axie Infinity (AXS)	AXSUSDT, AXSBTC, ETHUSDT
Exchange Tokens	Tokens issued by crypto exchanges for fee discounts, governance, or utility.	Binance Coin (BNB)	BNBUSDT, BNBBTC, ETHUSDT
Layer 1 & Layer 2	Core blockchains (Layer 1) and scaling solutions (Layer 2)	Ethereum (ETH), Polygon (MATIC)	ETHUSDT, ETHBTC, MATICUSDT

	that improve performance.		
Privacy Coins	Cryptocurrencies focused on anonymity and transaction privacy.	Monero (XMR)	XMRUSDT, XMRBTC
AI & Big Data	Projects integrating artificial intelligence and big data into blockchain applications.	SingularityNET (AGIX)	AGIXUSDT, AGIXBTC

These additional data sources are integrated in **Step 3** as part of the ELT process. In preparation for this integration, a placeholder field named `Category` was included in the relational schema to allow future enrichment without structural redesign.

To validate the correctness of the 3NF implementation, a UML/ER diagram was generated to logically represent the table structure and relationships.

UML Diagram



Preliminary results:

1. Technical Foundation & MySQL Setup

Prior to data processing, the local infrastructure was prepared.

- **MySQL Installation:** Installed on macOS via Homebrew.
- **Connection & Security:** Due to initial authentication errors (Error 1045), the local MySQL server was configured for development without a password (password: '').
- **Python Connectivity:** The `mysql-connector-python` package was used for communication between Python and the database (documented in requirements.txt).
- **Virtual Environment:** All Python dependencies are isolated in the `.venv` folder.

2. Generic API Function (from Step 1)

The first data extraction milestone was the creation of modular logic for retrieving market data (pairs).

- **Function:** `get_binance_data()` uses the `requests` library to query the Binance API endpoint `/api/v3/ticker/price`.

- **Modularity:** The function can retrieve either all available tickers or a specific symbol (e.g. BTCUSDT).

- **Error Handling:** Implemented try–except blocks to handle network errors or invalid API responses.

3. Relational Database Design (3NF) – Step 2

The goal of this milestone was to convert flat API data into **Third Normal Form (3NF)** to eliminate redundancy and ensure data integrity.

Script: 1_setup_database.sql

Development Note: At the beginning of the script, existing tables are dropped before being recreated. This ensures that structural changes (e.g. modified columns) are applied correctly during schema development, since `IF NOT EXISTS` prevents updates to existing tables. This logic is intended for development purposes and should be removed before final deployment.

Creation of the relational schema in the database `CryptoBot_Step2`.

The script defines a 3NF-compliant structure consisting of three core tables and enforces referential integrity through primary and foreign key constraints.

- **Table Currency (Master Data):** Stores unique currency and cryptocurrency identifiers (e.g. BTC, USDT) together with a descriptive attribute for asset type
    - Primary Key: `currency_id`
    - Ensures each currency is stored only once (no redundancy)
    - Stores: `asset_type` (crypto vs. stablecoin)

- **Table Pair (Structural Data):** Stores trading pairs (e.g. BTCUSDT), links them to their respective base and quote currencies, and adds a placeholder field for future category enrichment.
    - Primary Key: `pair_id`
    - Foreign Keys:
        - `base_currency_id` → references `Currency(currency_id)`
        - `quote_currency_id` → references `Currency(currency_id)`
    - Both foreign key columns are defined as **NOT NULL**, ensuring that every trading pair is linked to valid currency records and preventing incomplete pair definitions.
    - This guarantees strict 1:N relationships between Currency and Pair.

- **Table Price_Hist (Transactional Data):** Stores historical price observations with timestamps.
    - Primary Key: `price_hist_id`
    - Foreign Key:
        - `pair_id` → references `Pair(pair_id)`
    - `pair_id` is defined as **NOT NULL**, ensuring that every price entry is linked to a valid trading pair.

- Stores:
  - `price` (FLOAT / DECIMAL)
  - `timestamp` (generated at insertion time)
- This design ensures that transactional data remains separated from structural and master data in compliance with Third Normal Form.

Script: 2_ingest_binance_3nf.py

Automates the ETL process (Extract, Transform, Load):

- **Extraction:** Retrieves live prices for defined markets via API.
- **Transformation:** Splits trading symbols into currency components.
- **Loading:** Inserts data intelligently using `INSERT IGNORE` to prevent duplicate master data while continuously appending historical data.

Scripts: 3_check_data.sql & 4_functional_test.sql

- **Validation:** Simple queries to verify record counts.
- **Functional Test:** A complex JOIN across all three tables reconstructs the normalised data logically, proving correctness of the design.

4. Folder Structure (GitHub)

Files were logically organised by project phases:

- **Step1/** – Exploration and initial API scripts.
- **Step2/** – SQL setups and final relational ingestion script.
- **Root** – Central configuration files such as `requirements.txt` and `README.md`.

Technical Conclusion (Step 2)

The implementation of the relational data model and subsequent analytical transformation successfully extended the project from pure data extraction to a structured and scalable analytics architecture.

In **Step 2**, the semi-structured API responses were converted into a relational schema in **Third Normal Form (3NF)**. By separating master data (Currency), structural data (Pair), and transactional data (Price_History), redundancy was eliminated and referential integrity was enforced through primary and foreign key constraints. Validation queries and functional JOIN tests confirmed the correctness and logical consistency of the model.