TruLux Data Scraping Project Report

Prepared By:
Alex Lester
alexlester505@gmail.com

Date:
March 12th, 2025

Table of Contents

List of Figures i

1. Introduction 1

2. Data Acquisition 1

2.1 Tool Evaluation 1

2.1.1 Tool Comparison 1

2.1.2 Tool Selection 3

2.2 Data Scraping 3

2.3 Data Storage 4

3. Data Cleaning and Normalization 4

3.1 Strategy Generation 4

3.2 Implementation of Strategy 5

4. Data Analysis and Visualization 5

5. Database Schema 9

References 11

List of Figures

Figure 1: Histogram showing the distribution of watch caliber frequencies (vph) after data cleaning 6

Figure 2: Pie chart showing the distribution of movement types (e.g., Automatic, Handwound, Quartz) in the cleaned dataset 7

Figure 3: Histogram showing the distribution of power reserves (in hours) across watch calibers after data cleaning. 8

Figure 4: Scatter plot showing the relationship between power reserve (hours) and frequency (vph) of watch calibers after cleaning. 9

Figure 5: SQL schema for storing watch caliber data. 10

1. Introduction

The purpose of this project is to collect, clean, and analyze data on watch calibers from publicly available online sources, specifically focusing on WatchBase.com. This report outlines the process of using LLM's to generate Python-based web scraping techniques to gather approximately 1,000 watch caliber entries, capturing key data fields such as Caliber Name, Brand, Movement Type, Complications, Beat Rate, and Power Reserve. The project also explores the evaluation and selection of appropriate scraping tools, data cleaning and normalization techniques, and initial analysis to uncover patterns within the collected dataset. By leveraging modern scraping methods and data analysis tools, this project aims to create a structured, high-quality dataset that can serve as a foundation for further research and insights into watch movements.

2. Data Acquisition

2.1 Tool Evaluation

Below is a detailed list discussing the main advantages and disadvantages of each major web scraping tool currently available for data acquisition projects. This comparison focuses on both traditional scraping tools and modern LLM-friendly tools, considering their functionality, ease of use, flexibility, and suitability for different types of websites. An LLM was used to generate this list. The prompt provided to the LLM was as follows:

"I am working on a data scraping project and need to evaluate the best scraping tools available. Can you generate a detailed list of the most commonly used web scraping tools, including both traditional tools and modern LLM-friendly tools? For each tool, please provide a list of pros and cons that discuss "How easy or hard the tool is to use, whether it handles static or dynamic websites, whether it can be integrated with LLMs, its flexibility, scalability, and limitations for large data scraping projects" I need this to compare which tool would be most appropriate for scraping a large dataset (1,000 entries) from WatchBase.com."

2.1.1 Tool Comparison

1. BeautifulSoup + requests

Pros:

- Lightweight and easy to set up with minimal dependencies.
- Well-documented with many online tutorials and community support.
- Provides fine-grained control over HTML parsing and data extraction.
- Highly effective for static websites with consistent HTML structures.

- Can be combined easily with pandas for data storage and analysis.

Cons:

- Cannot handle dynamic content or websites that require JavaScript rendering.
- Requires manual effort to handle pagination, retries, and error handling.
- Lacks built-in scalability for scraping large datasets — needs additional setup for parallel processing.
- Not directly integrated with LLM workflows; requires manual prompt engineering for code assistance.

2. Selenium

Pros:

- Capable of automating web browsers and handling JavaScript-heavy, dynamic websites.
- Allows simulation of user interactions like clicks, form submissions, and navigation.
- Supports multiple browsers (Chrome, Firefox, etc.) for flexible testing and scraping.
- Well-supported with extensive community resources and plugins.

Cons:

- Relatively slow and resource-intensive compared to non-browser tools.
- Requires setup of browser drivers (e.g., ChromeDriver) and ongoing maintenance.
- Complex for simple scraping tasks where full browser automation is unnecessary.
- Not inherently LLM-friendly; requires additional prompting to generate Selenium-specific scripts.

3. Scrapy

Pros:

- High-performance, production-ready framework for large-scale and complex scraping projects.
- Built-in features for handling requests, pipelines, data storage, and error management.
- Designed for scalability and speed, including concurrent scraping and distributed crawls.
- Strong community and extensive documentation.

Cons:

- Steeper learning curve, especially for beginners.
- More rigid structure requiring custom spider definitions and pipelines.
- Limited built-in support for dynamic content — needs middleware like Splash for JavaScript rendering.
- Not inherently integrated with LLM tools, although code generation for spiders is possible with LLM assistance.

4. unclecode/crawl4ai (LLM-Friendly)

Pros:

- Allows users to define scraping tasks using natural language prompts, reducing the need for manual coding.
- Seamlessly integrates LLMs to generate scrapers, making it faster to start a project.
- Can simplify data extraction when site structure is well-aligned with LLM understanding.
- Potential for rapid prototyping and adjusting scraping targets with simple text updates.

Cons:

- Newer and less mature tool with limited documentation and community support.
- May struggle with complex or dynamically loaded content without clear cues.
- Less control over fine-grained parsing unless combined with manual adjustments.
- Performance and error handling may require extra attention for large datasets.

5. Playwright with LLM agents (e.g., AutoGPT, LangChain integrations)

Pros:

- Modern and powerful for handling dynamic, JavaScript-heavy websites.
- Supports headless and full browser automation for complex scraping tasks.
- Scriptable interactions (clicks, scrolling) and powerful control over page behavior.
- Can be integrated with LLMs to automate complex workflows and dynamic content scraping.

Cons:

- Requires more setup and configuration than simpler libraries.
- Higher resource consumption and slower performance compared to static scraping tools.
- Overkill for simple static websites or straightforward scraping needs.
- Still requires careful prompt engineering when used with LLMs to ensure correct script generation.

2.1.2 Tool Selection

Ultimately, considering the scope of this project and the fact that WatchBase.com is a semi-static website, the most effective and appropriate tools for the scraping component are BeautifulSoup and Requests. These libraries offer the necessary flexibility and simplicity to efficiently extract the required data without the overhead of more complex tools like Selenium or Scrapy. Additionally, BeautifulSoup and Requests offer greater control over the scraping process and avoid unnecessary reliance on AI-based tools, which may struggle with non-standard web structures or introduce complexity when scraping well-structured sites like WatchBase.com.

2.2 Data Scraping

Continuing to utilize the same LLM that generated the list found in the section above the following prompt was given to generate the python code for scraping the website:

“I need to scrape approximately 1,000 watch caliber entries from watchbase.com. I want to collect the following data fields: Caliber Name/Designation, Brand, Movement Type, Complications, Beat Rate and Power Reserve. Please write Python code using bs4 and requests to scrape this data and store it in CSV format. Also, ensure that the scraping randomizes or distributes the selection across brands and production years to avoid bias, and targets a representative sample of calibers if possible. Include any necessary imports, setup, and comments for clarity."

The code that the LLM generated was tested in a python file. The provided code was a good starting point to develop the necessary program however achieve the desired results the code had to be fixed, built upon and tested. The final version of the code for this project can be found in the provided python file titled, “trulux_project.py”.

2.3 Data Storage

The above prompt saved the raw data that was scraped in a CSV file that is labeled “raw_data.csv”. That CSV file is also provided with this report. Later after the raw data is cleaned and normalized it is then saved to another csv file called “cleaned_data.csv”.

3. Data Cleaning and Normalization

3.1 Strategy Generation

After the raw data is saved to the CSV file the data must then be cleaned and checked that there is no missing information. Utilizing an LLM I provided this prompt:

“I have scraped a dataset of 1,000 watch calibers from WatchBase, including fields like Name, Brand, Movement, Display, Reserve, Frequency, and URL. I want to clean and normalize this dataset to prepare it for analysis. Can you suggest a detailed data cleaning and normalization strategy, including methods for handling missing values, standardizing inconsistent formats (like numeric fields stored as text), removing duplicates, and fixing data quality issues?”

The LLM responded with the following strategy. To ensure the quality and usability of the scraped data, a comprehensive data cleaning and normalization process will be implemented. This process begins with identifying missing values in key fields such as Name, Brand, and Movement. If essential fields like Name and Brand are missing, the corresponding rows will be removed, while non-essential fields such as Reserve and Frequency will either be filled with placeholders like "Unknown" or left as NaN, depending on analysis needs. Next, data formats will be standardized by converting numeric fields like Reserve (hours) and Frequency (vph) to proper numeric types using pd.to_numeric(), handling any conversion errors gracefully. Leading and trailing spaces will be stripped from all string fields to ensure consistency, and units will be standardized where necessary. To address redundancy, exact duplicates will be removed using

drop_duplicates(), and potential logical duplicates will be checked based on combinations of key fields, such as Name and Brand. Additionally, text casing will be normalized—for example, by capitalizing brand names properly—and common placeholders like "n/a", "unknown", and "N/A" will be standardized to NaN. Non-informative characters, such as trailing periods or unnecessary special characters in names, will also be removed. Finally, a verification step will summarize the cleaned dataset to ensure that all fields are properly formatted and that critical fields are populated.

3.2 Implementation of Strategy

After the LLM provided its strategy, I prompted it to provide the code to execute it. The code that the LLM generated can be found in the python file that is provided with this report.

4. Data Analysis and Visualization

To visualize both the raw and cleaned data, I utilized Matplotlib due to its flexibility and effectiveness in creating clear and informative charts. Additionally, I chose Matplotlib because I have prior experience with this library, allowing me to efficiently generate the required visualizations. Again to speed up the development process I utilized an LLM to generate the necessary python code to accomplish the task.

“I have a dataset stored in a CSV file. The dataset includes fields such as Name, Brand, Movement, Display, Reserve, Frequency, and Complications. Can you generate Python code using Matplotlib to create the following visualizations:

1. A pie chart showing the distribution of different movement types (e.g., automatic, manual, quartz).
2. A bar chart showing the frequency of the most common complications found in the dataset (if available).
3. A histogram showing the distribution of beat rates (Frequency, in vph).
4. A histogram showing the distribution of power reserves (in hours).
5. A scatter plot to visualize the relationship between power reserve and frequency (to see if higher frequency movements correlate with higher or lower power reserves).”

The LLM responded with the necessary python code which can be found in the provided python file. The code itself needed some minor modifications however was mostly correct. I provided the LLM with some of the charts and asked it to analyze them and offer insights.

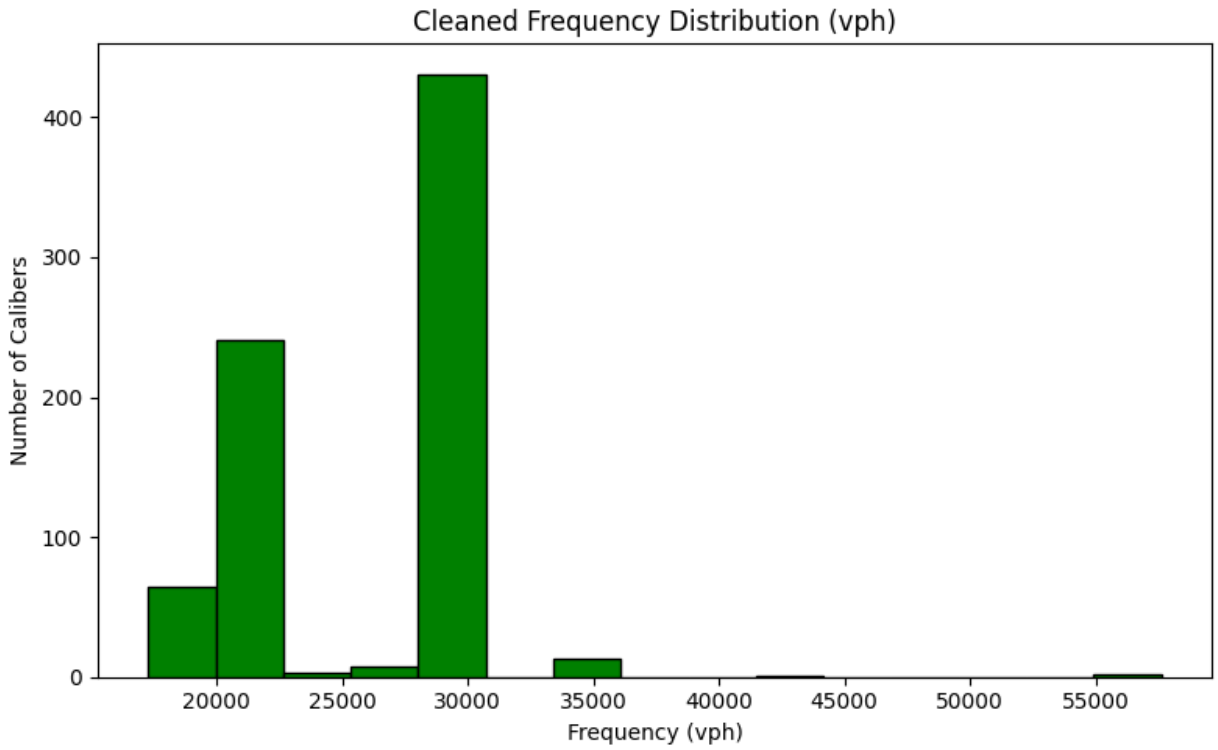

Figure 1: Histogram showing the distribution of watch caliber frequencies (vph) after data cleaning

This histogram illustrates that most watch calibers operate around common frequencies, with the highest concentration at 28,800 vph, followed by clusters around 21,600 vph and 18,000 vph. These frequencies are standard in many modern mechanical watches, reflecting industry norms for balance wheel oscillation rates. Some outliers exist at higher frequencies, but these are far less common.

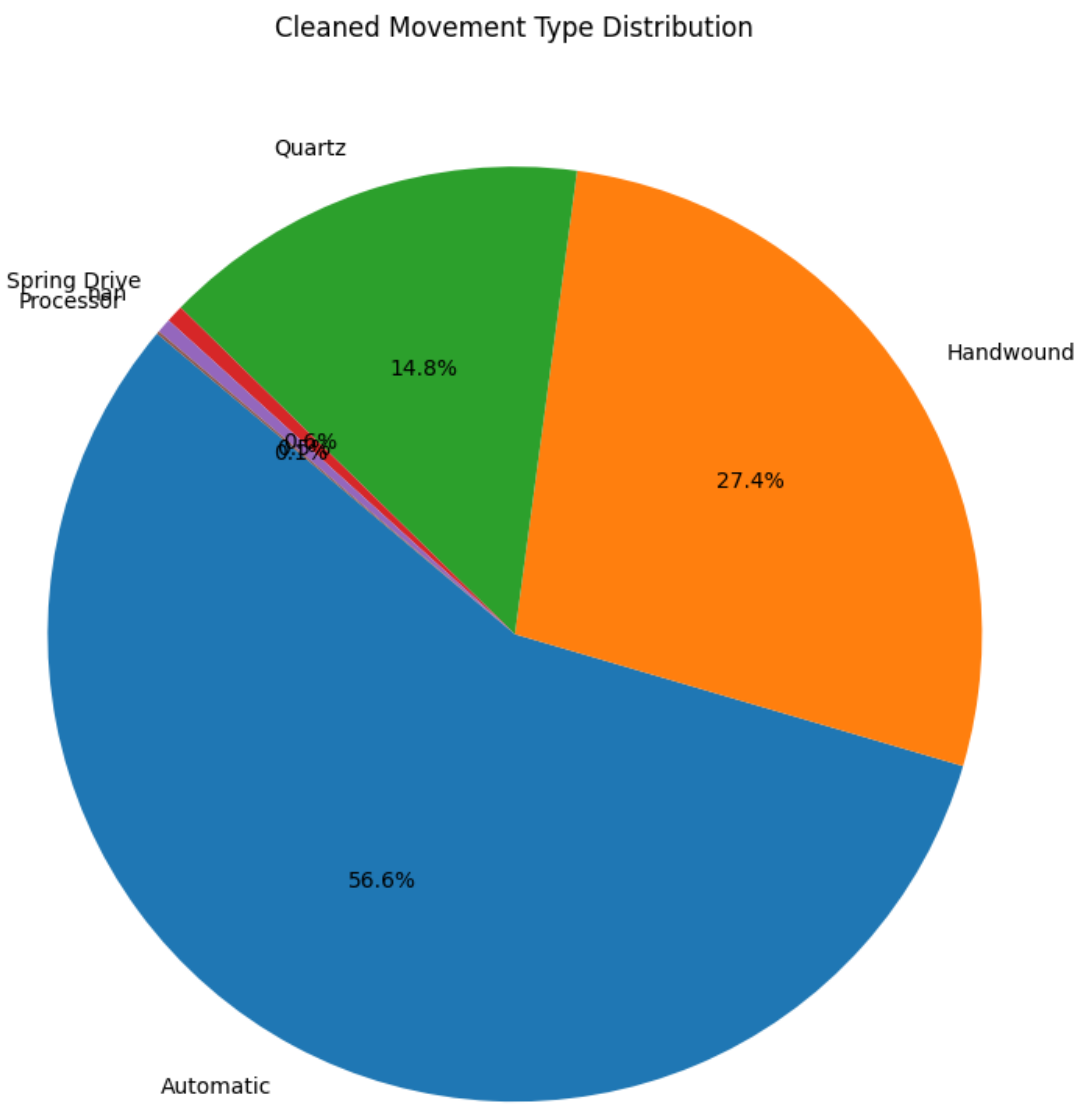

Figure 2: Pie chart showing the distribution of movement types (e.g., Automatic, Handwound, Quartz) in the cleaned dataset

This pie chart shows that Automatic movements are the most prevalent, comprising 56.6% of the dataset, followed by Handwound (27.4%) and Quartz (14.8%). Other types like Spring Drive, Processors, Solar make up a very small fraction of the total. This distribution reflects a strong preference for traditional mechanical movements in high-end calibers.

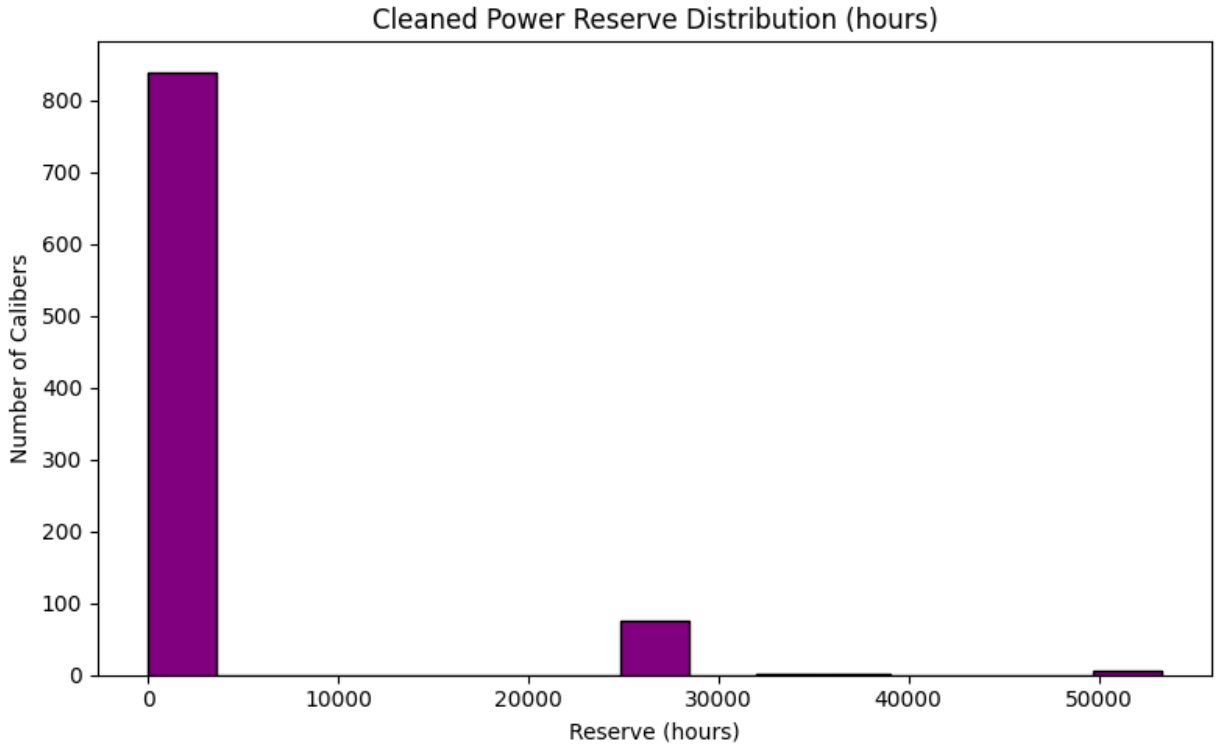

Figure 3: Histogram showing the distribution of power reserves (in hours) across watch calibers after data cleaning.

Most watch calibers have a power reserve under 100 hours, which is standard for mechanical watches. However, there are a few extreme outliers reaching up to 50,000 hours, likely representing perpetual calendar types or errors needing review. The clustering near lower reserves indicates that most movements are designed for daily wear.

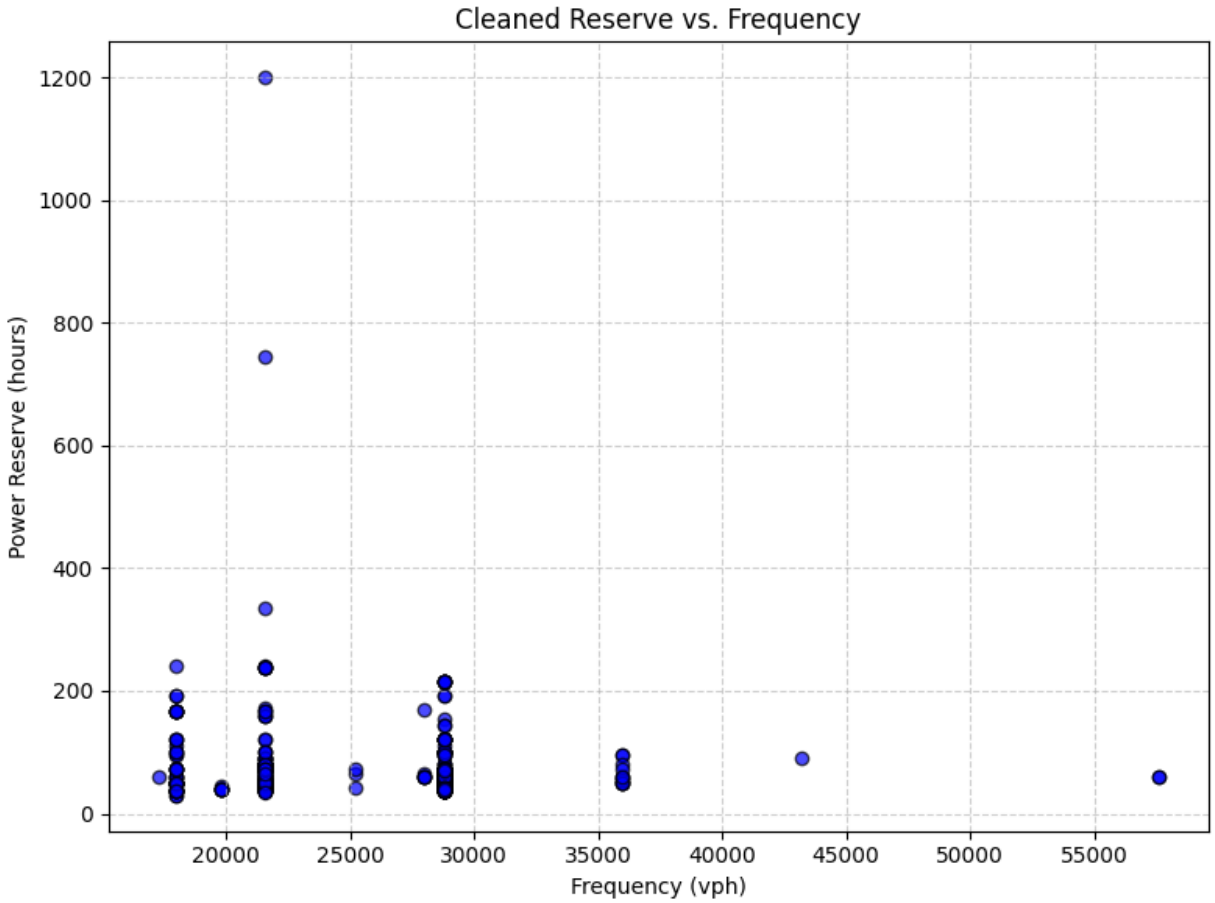

Figure 4: Scatter plot showing the relationship between power reserve (hours) and frequency (vph) of watch calibers after cleaning.

This scatter plot reveals no clear linear relationship between frequency and power reserve, suggesting that while higher frequency might impact energy usage, it doesn't uniformly determine reserve length. Notably, some calibers with standard frequencies like 28,800 vph still achieve high power reserves, indicating innovations in energy efficiency and barrel design. Outliers with extreme reserves may reflect special engineering solutions or errors.

5. Database Schema

For this section of the project this prompt was given: “Based on my scraped watch caliber dataset, which includes fields like Name, Brand, Movement, Display, Reserve, Frequency, Complications, and URL, should I store this dataset using an SQL or NoSQL database? Please justify the choice based on data characteristics and anticipated query patterns. Then, provide a detailed schema including table/collection names, field names, data types, and relationships. If you recommend SQL, include CREATE TABLE statements. If you recommend NoSQL, describe the document structure.”

LLM Response:
Considering the nature of the watch caliber dataset — which consists of structured, relational data with consistent fields such as Brand, Movement, and numeric characteristics like Reserve and Frequency — a SQL (relational) database is recommended. SQL databases are better suited for datasets that are highly structured, where you may want to perform complex queries, filters, joins. As such, SQL (e.g., SQLite, PostgreSQL, or MySQL) is recommended over NoSQL.

```sql
-- Main table to store watch caliber data
CREATE TABLE calibers (
    id INTEGER PRIMARY KEY AUTOINCREMENT, -- Auto-increment ID for internal referen
    name TEXT NOT NULL,                    -- Caliber name/designation
    brand TEXT NOT NULL,                   -- Brand name
    movement TEXT,                         -- Movement type (e.g., Automatic, Manual,
    display TEXT,                          -- Display type (e.g., Analog, Digital)
    reserve INTEGER,                       -- Power reserve in hours
    frequency INTEGER,                     -- Frequency in vph (vibrations per hour)
    complications TEXT,                    -- List of complications, stored as a text
    url TEXT                               -- URL to the caliber detail page
);
```

Figure 5: SQL schema for storing watch caliber data.

References

[1] "ChatGPT," OpenAI, [Online]. Available: https://chatgpt.com/.

[2] "WatchBase," WatchBase, [Online]. Available: https://watchbase.com/calibers. [Accessed 2025].