**Data Collection:**  
Calculating the Bitcoin MVRV (Market Value to Realized Value) ratio will need two main kinds of information: the market information and the on-chain transaction information. Current market data can be freely retrieved with free public APIs like CoinGecko; data that includes the current market price, circulating supply, and market capitalization of Bitcoin can be easily downloaded. The APIs enable developers to access real-time values through easy requests and thus are best suited to develop automation and calculate MVRV values at fixed intervals, such as every hour or day.

The calculated realized capitalization needs on-chain transaction data. The process can be done on the UTXO (Unspent Transaction Output) set, in which its respective outputs are pegged to the value of a Bitcoin at the last point of movement. To find this information, developers can use free blockchain explorer APIs such as Blockchair or SoChain, which contain transaction history and UTXO information. It is also possible to fetch historical price data on these timestamps on the CoinGecko market\_chart/range endpoint. Collectively, such an integration of real-time market data and historical transaction data allows tracking the MVRV ratio in real-time and maintains precision in the process without necessarily using paid services.  
  
**Data Storage:**  
We are going to employ the use of SQL (Structured Query Language) this example of an MVRV Ratio project in order to store our data given that it is a quality, organized and effective mode in which to handle the various classifications of data applied to this project, notably time-series, transactional and calculated values. SQL-based technologies, such as PostgreSQL, perform well in dealing with the relatively structured nature of our data, including points of price changes in Bitcoin, UTXO records, and calculated MVRV ratios, which are time-stamped. SQL also offers robust querying features, and it is easy to query, accumulate, and analyze the data across time (e.g., hourly or daily). SQL also protects data validity, taking into consideration points such as indexing, constraints, and connecting tables to one another.

In this project, we will organize our database with three across-but-related tables: one for storing variables based on hourly price and circulating supply, one table to monitor snapshots of UTXO data and their historical values, one table to add historic prices of Bitcoin, and the last table is to hold the computed values of MVRV ratios. This relational model enables us to have clean normalized data and it lets us have efficient joins and queries between sets.

price\_data

('hourly'/'daily')

(computed using historical price)

utxo\_data

Id

Txd

Value\_btc

Moved\_timestamp

Value\_usd

mvrv\_ratios

Id

Timestamp

Market\_cap

Ratio

timeframe

Id

Timestamp

Price\_usd

Historical\_prices

Id

times tamp

price\_usd

supply

With the help of foreign keys and unique configs, we will be sure that every entry of data will be internally consistent and that historical pricelet references will be accurately associated with their appropriate UTXOs. Being older and more compatible with the tools, SQL will be a great option to go and generate a stable and transparent data pipeline to accommodate the end-to-end life cycle of Bitcoin MVRV calculation through libraries like psycopg2 or SQLAlchemy.  
  
**Schema Design**

1. **price\_data**

Stores hourly Bitcoin market price and circulating supply.

*CREATE TABLE price\_data (*

*id SERIAL PRIMARY KEY,*

*timestamp TIMESTAMP UNIQUE NOT NULL,*

*price\_usd NUMERIC NOT NULL,*

*supply NUMERIC NOT NULL*

*);*

1. **historical\_prices**

Stores BTC price at specific timestamps (used for UTXO valuation).

*CREATE TABLE historical\_prices (*

*id SERIAL PRIMARY KEY,*

*timestamp TIMESTAMP UNIQUE NOT NULL,*

*price\_usd NUMERIC NOT NULL*

*);*

1. **utxo\_data**

Stores unspent transaction outputs with historical value.

*CREATE TABLE utxo\_data (*

*id SERIAL PRIMARY KEY,*

*txid TEXT NOT NULL,*

*value\_btc NUMERIC NOT NULL,*

*moved\_timestamp TIMESTAMP NOT NULL,*

*value\_usd NUMERIC, -- computed using historical\_prices*

*FOREIGN KEY (moved\_timestamp) REFERENCES historical\_prices(timestamp)*

);

1. **mvrv\_ratios**

Stores calculated MVRV values.

*CREATE TABLE mvrv\_ratios (*

*id SERIAL PRIMARY KEY,*

*timestamp TIMESTAMP UNIQUE NOT NULL,*

*market\_cap NUMERIC NOT NULL,*

*realized\_cap NUMERIC NOT NULL,*

*ratio NUMERIC NOT NULL,*

*timeframe TEXT CHECK (timeframe IN ('hourly', 'daily'))*

*);*

**Calculation of MVRV Ratio:**MVRV (Market Value to Realized Value) ratio is computed as the ratio of the present Market Capitalization to Realized Capitalization of any cryptocurrency. The metric gives an understanding of whether the asset is overpriced or underpriced compared to the average cost basis of the investors. High MVRV suggests a top in the market as the asset has been well above the price of its purchase, whereas low MVRV may imply undervaluation.

The Market capitalization is obtained by multiplying the current price of Bitcoin (in USD) by the circulating supply at the time of consideration. It is obtained by retrieving a source such as CoinGecko. Instead, the Realized Capitalization is more sophisticated and consists of analyzing the UTXO (Unspent Transaction Output) set. Every UTXO acts as a piece of Bitcoin that has not been exchanged and is dependent on the previous transfer. For each UTXO, we calculate its value according to the last movement, which is the number of bitcoins in the UTXO multiplied by the value of Bitcoin at the time of movement (obtained through a historical price API such as the CoinGecko range endpoint). We add up these individual UTXO values, and that gives the total realized Capitalization. Now to finish, we take the Market Capitalization / Realized Capitalization and end up with the MVRV Ratio.

**Processing Frequency:**

The system is recommended to calculate the values of the MVRV ratio of Bitcoin on a time basis, and the results are suggested to be collected daily; this would guarantee the timely and accurate tracking. This method equalizes requirements for such insights within near real-time, with the monitoring of trends over the long term. The system follows a main set of two different schedules: an hourly job that fetches and computes data once every hour (at the top of the hour), and a daily job that summarises these results once every day (e.g., at midnight UTC).

The hourly procedure is retrieving the latest Bitcoin price and supply from an API such as CoinGecko, any new added UTXOs to the blockchain explorer (e.g., Blockchair), and their final movement date. To obtain the historical price of Bitcoin at a given timestamp, the system retrieves a historical price provided by a stored cache or by the historical price API provided by CoinGecko (one price per UTXO). Based on this information, the system will then compute the current Market Capitalization, Realized Capitalization, and the resulting MVRV ratio, which will then be inserted into the mvrv\_ratios table with a timeframe = 'hourly'.

The aggregation process is performed once a day, and it creates a daily total with the hourly records it has stored. It can be accomplished by averaging the result, getting the highest and lowest, and just taking the final recorded hourly MVRV ratio on the day, depending on analytical requirements. The outcome is stored in the mvrv\_ratios table of timeframe = daily. Such a dual-frequency system provides not only a fine-grained visibility but also a long-term historical view, and a scalable and organized structure that can be utilized in visualization, reporting, and detection of anomalies.

​ **Presentation Layer:**For the presentation part of the MVRV Ratio system, we chose Streamlit because of its simplicity and flexibility, as well as its good integration with Python. Streamlit enables us to develop a lightweight interactive dashboard without having to use an intricate front-end structure. It can be used perfectly with projects of this kind, where the priority is rapid visualisation of time-series financial data and a clean interface for users, i.e., one of those analysts, students, or crypto enthusiasts.

The hourly MVRV ratio and daily MVRV ratio will be posted on the dashboard in interactive charts, tables, and filters. The user may choose a range of dates to see the past trends or look at the latest hourly fluctuations. Important data such as the current price, market capitalization, realized capitalization, and MVRV ratio will be displayed in real-time. There are Python packages such as Pandas to manipulate data, and Matplotlib or Plotly to plot in Streamlit. One more thing we can add is the drop-down menus to choose the time period, auto update data margins, and conditional formatting to show overestimated or underestimated areas. All in all, Streamlit makes the system accessible, simple to manage and deploy, and readily prepared to be iterated on.