Market tracker

**🗃️ Database: cs2\_market**

This is your main database that stores all Counter-Strike 2 item market information.

It contains the following key collections:

**1️⃣ Collection: all\_item\_names**

**Purpose:**  
Your **reference table** for all known CS2 market items — essentially your “master index” of every item that has ever appeared on the market.

**Used for:**

* Searching items by keywords or names.
* Joining (linking) with other tables via csfloat\_id.
* Detecting **newly added items** from CSFloat daily snapshots.
* Providing human-readable item names to your front end or API.

**Example Document:**

{

"\_id": ObjectId("67003af0..."),

"csfloat\_id": "AWP | Duality",

"raw\_name": "AWP | Duality",

"keywords": ["awp", "duality"],

"created\_at": "2025-10-04T19:20:00Z",

"last\_seen": "2025-10-04T19:20:00Z"

}

**Key points:**

* ✅ **csfloat\_id** = the stable unique key (usually the item’s item\_name).
* ✅ Used to link to csfloat\_price\_history and other collections.
* ✅ Searchable by **keywords**.
* ✅ New items are automatically inserted by your **daily snapshot script**.

**2️⃣ Collection: csfloat\_daily\_prices**

**Purpose:**  
This is your **temporary snapshot table** — it stores raw data fetched directly from the **CSFloat API** for a specific day.

Think of it as a staging area or “daily feed” before processing.

**Used for:**

* Holding today’s API data.
* Feeding the daily snapshot process that updates historical prices.
* Detecting new items not yet in all\_item\_names.

**Example Document:**

{

"\_id": ObjectId("67003b91..."),

"date": "2025-10-04T00:00:00Z",

"item\_name": "AWP | Duality",

"condition": null,

"price": 1234,

"quantity": 12,

"fetched\_at": "2025-10-04T19:17:12Z",

"raw\_cs": {

"market\_hash\_name": "AWP | Duality (Factory New)",

"qty": 12,

"min\_price": 1234

}

}

**Key points:**

* ✅ Updated **daily** using CSFloat API (/api/v1/listings/price-list).
* ✅ Serves as input for the **daily snapshot script**.
* ✅ Cleared or refreshed each day after being processed (optional but recommended).
* ⚙️ Transient — not meant for long-term storage.

**3️⃣ Collection: csfloat\_price\_history**

**Purpose:**  
This is your **long-term historical table** — it keeps **all daily price and quantity changes** over time in a compact format.

Each document represents **one item**, with daily data stored in dictionaries keyed by date.

**Used for:**

* Analyzing price trends and item volatility.
* Showing historical charts or average price movement.
* Detecting sudden market changes.
* Feeding machine learning or forecasting models later.

**Example Document:**

{

"\_id": ObjectId("67003c02..."),

"csfloat\_id": "AWP | Duality",

"daily\_prices": {

"2025-10-04": 1234,

"2025-10-05": 1225,

"2025-10-06": 1250

},

"daily\_quantity": {

"2025-10-04": 12,

"2025-10-05": 11,

"2025-10-06": 14

},

"last\_fetched\_at": "2025-10-06T19:00:00Z"

}

**Key points:**

* ✅ One document per item.
* ✅ Compact format — just one record per item with daily data nested inside.
* ✅ Grows efficiently (adds one key per day instead of one document per day).
* ✅ Easily joinable with all\_item\_names via csfloat\_id.

**4️⃣ (Optional) Collection: current\_prices**

**Purpose:**  
If you decide to track **real-time Steam prices** separately from CSFloat data, this can store your **latest Steam market snapshot**.

*(You already created this earlier for Steam’s priceoverview API.)*

**Used for:**

* Quick lookup of current market prices.
* Comparing CSFloat min prices vs Steam prices.

**Example Document:**

{

"\_id": ObjectId("67003c22..."),

"item\_name": "AWP | Duality",

"lowest\_price": 12.75,

"median\_price": 13.20,

"lowest\_price\_display": "$12.75",

"median\_price\_display": "$13.20",

"timestamp": "2025-10-04T18:19:16Z",

"source": "steam\_market\_priceoverview"

}

**Key points:**

* ✅ Stores live prices queried from Steam.
* ✅ Updated only when necessary (not daily).
* ✅ Used for cross-verification and alerting when price changes exceed a threshold.

**🧩 How They Work Together**

| **Step** | **Process** | **Involved Collections** |
| --- | --- | --- |
| 1 | Fetch daily CSFloat prices via API | csfloat\_daily\_prices |
| 2 | Update all known items (adds new ones) | all\_item\_names |
| 3 | Append or update daily prices | csfloat\_price\_history |
| 4 | Optionally fetch current Steam prices | current\_prices |
| 5 | (Optional) Compare Steam vs CSFloat data for anomalies | joins all collections |

**Example Data Flow (Daily Pipeline)**

CSFloat API → csfloat\_daily\_prices

↓

update\_all\_item\_names.py

↓

csfloat\_price\_history\_daily\_dict.py

↓

all\_item\_names (updated)

csfloat\_price\_history (grows)

✅ **Result:**  
You now have a fully normalized, efficient, and scalable MongoDB setup that:

* Stores **raw daily data** (csfloat\_daily\_prices)
* Maintains a **clean item index** (all\_item\_names)
* Builds a **lightweight historical record** (csfloat\_price\_history)
* Can optionally include **live market data** (current\_prices)