Imagine you're generating a report like this:A computer screen shot of a black screen

AI-generated content may be incorrect.

A screen shot of a computer program

AI-generated content may be incorrect.

Seems straightforward — until you want to **check if the same report was already generated**.

**😬 The Matching Nightmare**

You now need to query your DB like this: to find duplicate or to reuse already generated report.

A screen shot of a computer program

AI-generated content may be incorrect.

That’s:

* In-memory filtering
* Complex LINQ logic
* No real indexing
* Slower as data grows

**✅ The Better Way: Use a Hash**

Instead of comparing all 15+ parameters each time, generate a **deterministic hash** like this:

A screenshot of a computer program

AI-generated content may be incorrect.

Store the hash with each request.

Now your query becomes:A screen shot of a computer code

AI-generated content may be incorrect.

⚡ Fast  
⚡ Indexable  
⚡ Simple

💡 **If you're matching on many fields or rows — whether it's reports, payments, scheduling, or config — hashing the input can dramatically simplify your logic.**

🔁 Stop comparing all the fields. Start comparing the hash.

Hashing provider which I used XXHash,

A screen shot of a computer program

AI-generated content may be incorrect.

| **Use Case** | **Best Choice** | **Output Length** | **Notes** |
| --- | --- | --- | --- |
| Cryptographic security | SHA256 (full) | 64 hex / 44 b64 | Strong, but large |
| Fast + compact fingerprint | CRC32 or XXHash | 8–10 chars | Fast, good for duplicates |
| Safe + moderately short | Truncated SHA256 | 8–16 chars | Low collision chance, compact |
| Compact + alphanumeric | Base62(SHA256) | ~22 chars | Very short and safe if encoded well |