**Code1 :**

Hey, I’m working on merging two DataFrames in pandas. I’m a DH student and we’ve covered it before — like I remember how merge() works, and that I can specify on and how. But now I’m a bit stuck. I’ve got one DataFrame with pairs of filenames and a similarity score (filename-1, filename-2, similarity), and another one that has topic labels for individual files (file, topic). I want to add topic info for both filenames in the similarity DataFrame, but I’m getting confused about how to do the merge twice without messing it up. Don't give me solutions just guide me

2/2

**ChatGPT said:**

Got it! Since you already understand merge() and the basics, let’s walk through the thought process together instead of handing you code.

You have two DataFrames:

1. **Similarity DataFrame** with columns:  
   filename\_1, filename\_2, similarity
2. **Topics DataFrame** with columns:  
   file, topic

Your goal is to **add the topic label for both filename\_1 and filename\_2** into the similarity DataFrame.

So think of this like doing **two separate lookups** from the topics DataFrame — one for filename\_1, one for filename\_2.

Here are some guiding questions to help you plan:

* 💡 First: When you merge for the first time (say for filename\_1), what do you want to name the new topic column that comes in? Will you rename it right after the first merge so the second merge doesn’t overwrite it?
* 🔁 Second: When you do the second merge for filename\_2, how will you avoid conflicts with the already-merged file and topic columns from the first merge?
* 🧼 Lastly: How will you clean up any column name duplication (file\_x, file\_y) or drop any extra columns you don’t need?

Try writing down what columns you'd expect after each merge — it can help make sure you don’t overwrite or duplicate anything by mistake.

Want to sketch your plan here and I’ll walk through it with you?

**You said:**

But the column names are different. rename one of them?

**ChatGPT said:**

Exactly — you're on the right track! Since filename\_1 and filename\_2 are both meant to match against the same file column in the topics DataFrame, you need to **rename** the file column *before* each merge to avoid confusion.

Here’s the way to reason through it:

1. ✅ **First merge**:
   * Rename the file column in the topics DataFrame to something like filename\_1 **(just temporarily)**.
   * Merge on filename\_1.
   * You’ll get a topic column — consider renaming it to topic\_1 right after the merge.
2. 🔁 **Second merge**:
   * Repeat, but now rename file to filename\_2 before merging.
   * Merge on filename\_2.
   * Again, rename topic to topic\_2.

By doing this, you avoid:

* Matching the wrong columns,
* Accidentally overwriting the topic column from the first merge.

Would you like to test the first merge together just to see how it feels?

**You said:**

Also, i want to keep all rows from the left DataFrame (TF-IDF), and bring in matches from the right DataFrame (topics) if there any like this. what to do for that

**ChatGPT said:**

Great question! You want to **keep all rows from the left DataFrame (your similarity DataFrame)** no matter what, and only add topic info if there’s a matching file in the topics DataFrame.

In pandas merge terms, that means you want to do a **left join** (how='left').

So when you do each merge:

* Use how='left' to keep all similarity rows.
* The topic columns will be filled with the matching data if available.
* If there’s no matching topic for a filename, you’ll get NaN in the new topic column for that row.

Putting it together, your merge would look like:

python

CopyEdit

merged\_df = similarity\_df.merge(topics\_df\_renamed, left\_on='filename\_1', right\_on='filename\_1', how='left')

… and similarly for filename\_2.

Does that make sense? Want to talk through what happens to rows with no topic match?

**Code2:**

Also I want to have my dataset not the temporal info but only the filename 1 the topic 1 and then filename 2 then topic-2 then the similarity score. I can arrange it like that? do don't give me code but guide me

**ChatGPT said:**

Absolutely! Your goal is to have a clean, well-structured DataFrame with just these columns in this order:

* filename\_1
* topic\_1
* filename\_2
* topic\_2
* similarity

Here’s how to think about it:

1. After your two merges, your DataFrame will have some extra columns — like the original filename\_1 and filename\_2 from the similarity data, the two topic columns you added, and probably some leftover or duplicated columns like the file column from the topics DataFrame(s).
2. Your job is to **select just the columns you want** to keep, and **put them in the order you want**.
3. So, ask yourself:
   * Which columns do I need to keep?
   * What names do these columns have now?
   * How can I reorder them?
   * Are there any columns I want to drop to keep the DataFrame clean?
4. Remember, you can do this by:
   * Using a list of the exact columns you want and passing it to the DataFrame (e.g., df = df[[col1, col2, ...]])
   * Or by dropping unwanted columns (less recommended if you want precise control)
5. If you renamed the topic columns after merging (e.g., to topic\_1 and topic\_2), just pick those along with the original filenames and similarity score.

By keeping this mindset, you avoid having confusing or redundant columns, and your final DataFrame is clear and easy to use.

Want to try drafting which columns you expect after each merge and how you’d pick them? I can help review!

**You said:**

on printing i see alot of NANs, how to remove the rows that have these values and only keep those which have actual value

**ChatGPT said:**

Nice observation! Those NaNs usually pop up because:

* Some files in your similarity DataFrame didn’t find a matching topic in the topics DataFrame during the merge.
* So the topic columns for those rows stay empty (NaN).

If you want to **keep only rows where both topic\_1 and topic\_2 have actual values (no NaNs)**, here’s how to think about it conceptually:

* You want to **drop rows where either topic\_1 or topic\_2 is missing**.
* So, filter your DataFrame to **keep only rows where neither topic\_1 nor topic\_2 is null**.

In pandas terms, this means:

* Use a method that checks for missing values (like .notnull() or .dropna()) on those columns.
* Keep rows where both columns have real topic values.

This will make your DataFrame only contain rows with topics successfully matched for **both filenames**.

Does that make sense? Want to try describing in your own words how you’d do that?