# **Elasticsearch use case**

* Table example:
* See attached document.
* Note: USID – Is our internal Unique ID.
* How search should work:
* User enters a search query on website which will make an API call to Elasticsearch.
* Elasticsearch will take the query, find as many matches as possible.
* If the result has multiple products with similar names/characteristics, then those products will be aggregated into one.
* But, the product with the most information will be used as the base product. The additional match(es) to the base product will append some of the information to it into new columns in the base product.
* The extra columns to be added to the main row would be the store\_name\_x and product\_price\_x. Where x stands for the number of extra rows that have a 90% match.
* For example:
* Table has multiple rows of results of iPads with different information stored. Example in the table.
* User searches for iPad 2019 256gb gold.
  + First result should be iPad 2019 256gb gold
    - Gives two matches in the elasticsearch : USID 243002 & USID 267949. The first USID has the most information populated.
    - The result shown to the user should contain USID 243002 as the main row/result due to it having more information. This row should also contain two extra columns called store\_name\_2 and product\_price\_2 as there is one direct match with USID 267949. The values of these two rows should be the value of store\_name & product\_price\_1 for the match: USID 267949
    - Therefore, columns for USID 243002 will remain the same as it is the one with the most information, however two extra columns will be appended to it with the information from USID 267949.
  + Second result should be iPad 2019 128gb
    - E.G. USID 247696 and USID 267948 aggregated as above.
  + Third result should be iPad 2019 256gb
  + Fourth result should be iPad 2018 64 gb
  + Fifth result should be iPad 2018 128gb
  + Sixth result should be iPad 2018 256gb
  + Example results attached as ipad\_results
* These result values are stored as a separate ID in Elasticsearch. Therefore, next time someone searches for iPad 2019, these results will be automatically shown without it doing all the aggregation done above.
* Logstash will update elasticsearch constantly therefore the elasticsearch IDs will have to be restarted whenever there is new data available in elasticsearch.
* Note, this is based on a user search. However, if Elasticsearch is able to look at the names and automatically find similar matches and automate the process to store it in a separate table/different IDs then that would be more ideal. If Elasticsearch is not ideal for this, what would you recommend in using?
* Examples can also include other electronics including laptops, tv’s, phones. Can also include groceries from different places. For example, finding a match of 1L Milk from store X and a store Y