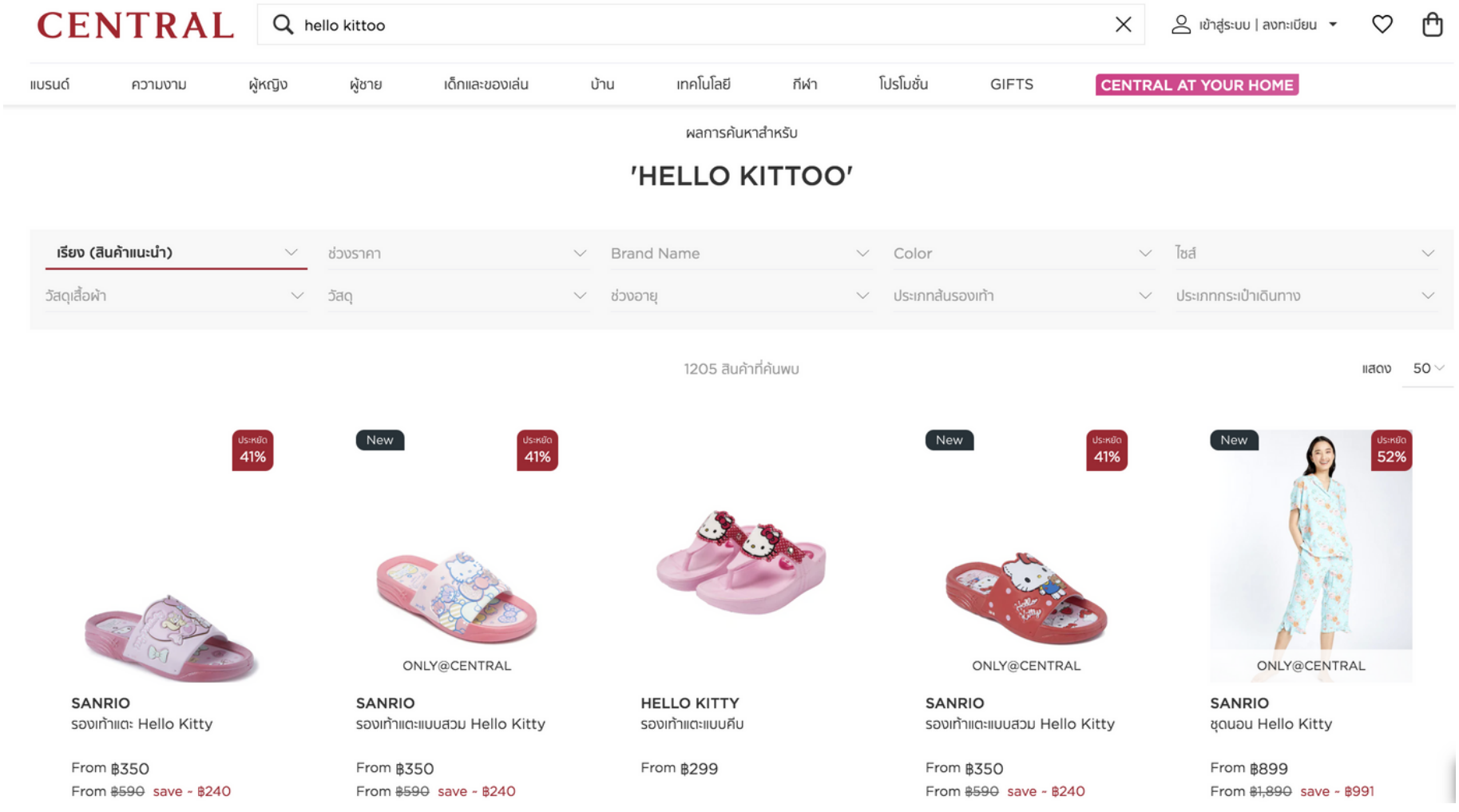
ENTER ELASTIC NINJA

Product Search and More with Elastic Stack

Charin Polpanumas
Lead Data Scientist @ Central

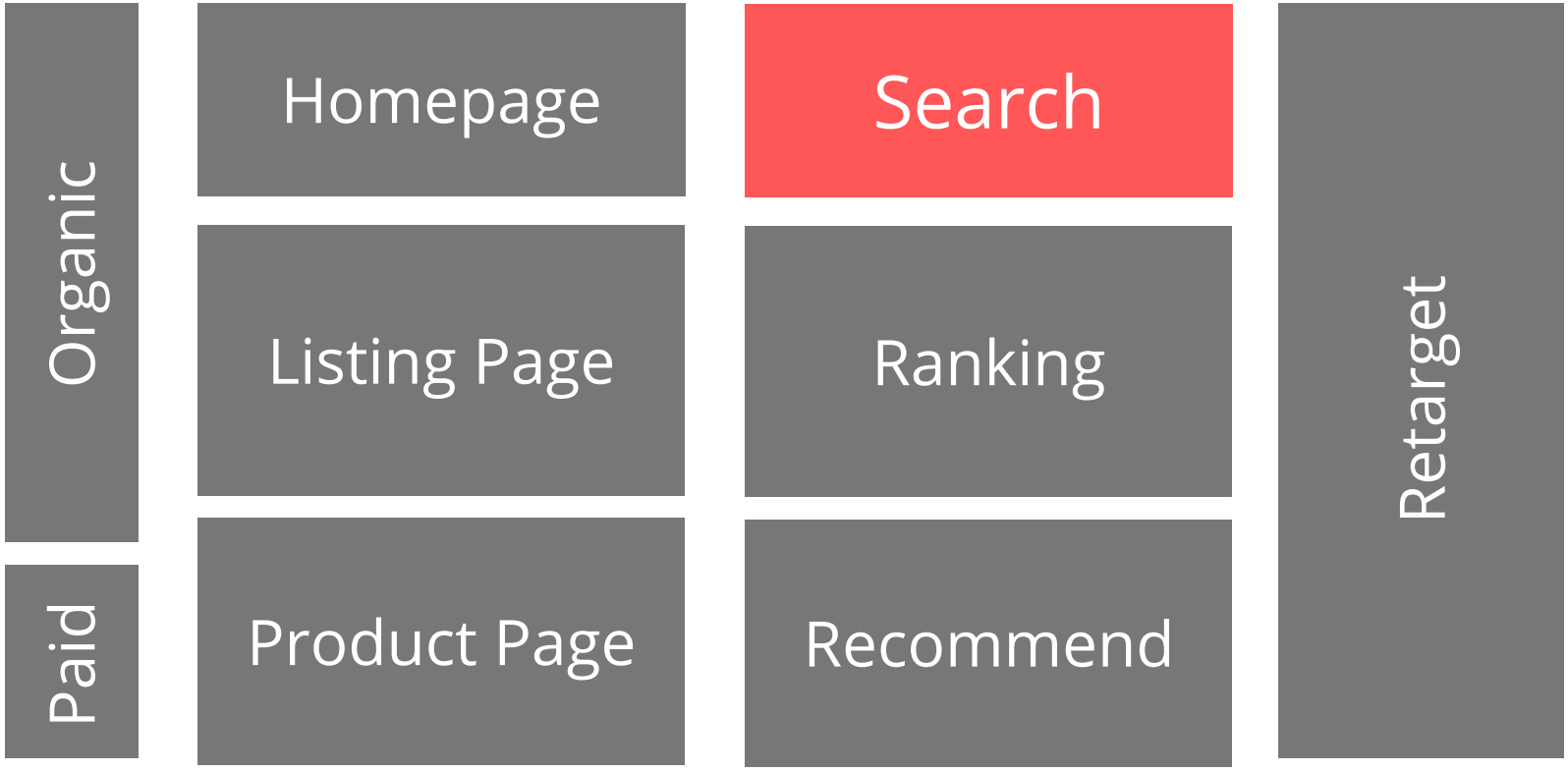
Product search in one sentence

Give a search term, show the products that user will most likely buy



Why people care about product search

Search is 10-20% of traffic but often has 5-10x conversion rate vs average



Layers of product search

What we will specifically cover today

Business Logic

You can find `chanel` on Central but not Robinson because of an exclusive deal

Re-ranking Model

Models that re-rank full-text search results to optimize for CTR/CR

Full-text Search

Match products purely based on texts of names, brand names, categories and other possible metadata

Why use Elasticsearch instead of regex or sklearn

Distributed, open source search and analytics engine for all types of data

Expression

```
/([A-Z])\w+/g
```

Text | Tests NEW

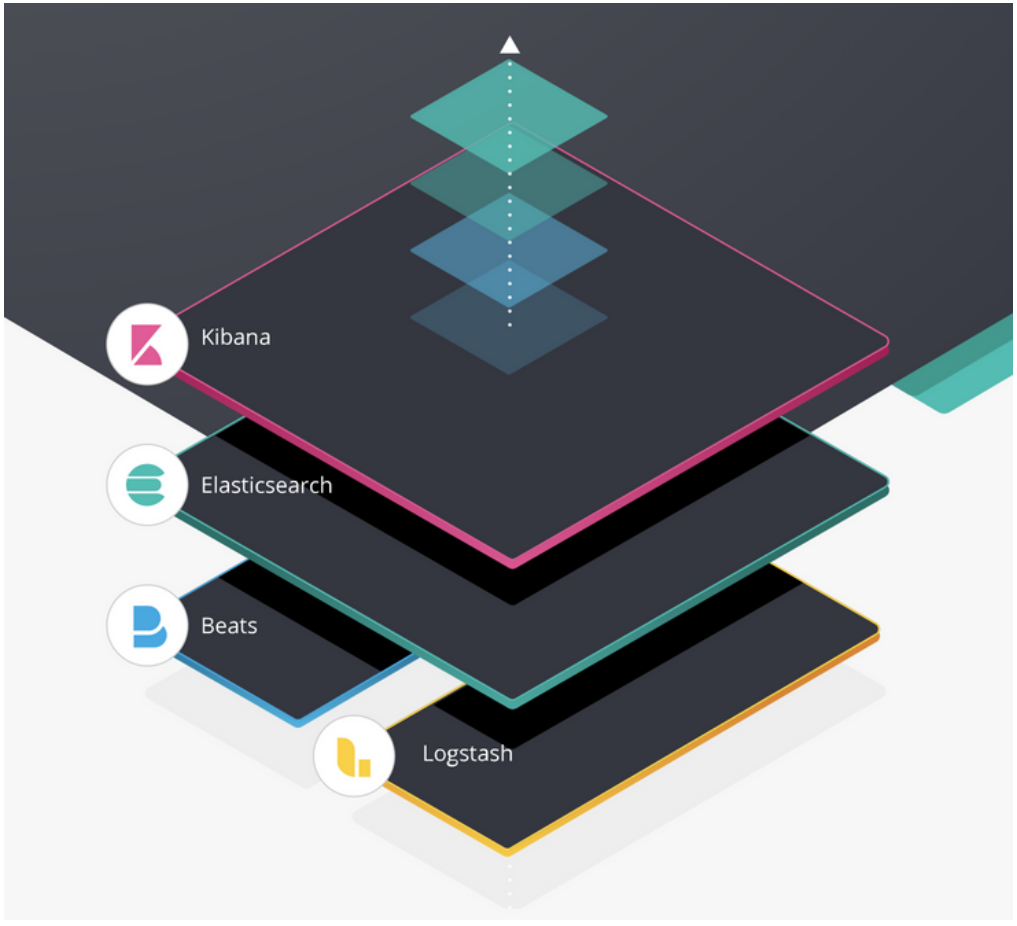
```
RegExr was created by gskinner.c

Edit the Expression & Text to se
are supported. Validate your exp

The side bar includes a Cheatshe
create or favorite in My Pattern

Explore results with the Tools b
expression in plain English.
```

```
from sklearn.feature_extraction.text import
corpus = [
    'This is the first document.',
    'This document is the second document.'
    'And this is the third one.',
    'Is this the first document?',
]
vectorizer = TfidfVectorizer()
X = vectorizer.fit_transform(corpus)
print(vectorizer.get_feature_names())
nd', 'document', 'first', 'is', 'one', 'seco
print(X.shape)
9)
```

- Natively Distributed; built on Apache Lucene
- Fast
- Scalable
- Resilient
- Rich in features
- Platform independent with REST APIs
- Excellent documentation and ease of use
- Largest community of any full-text search software

What is the Elastic Stack

Logstash and Beat, Elasticsearch, Kibana



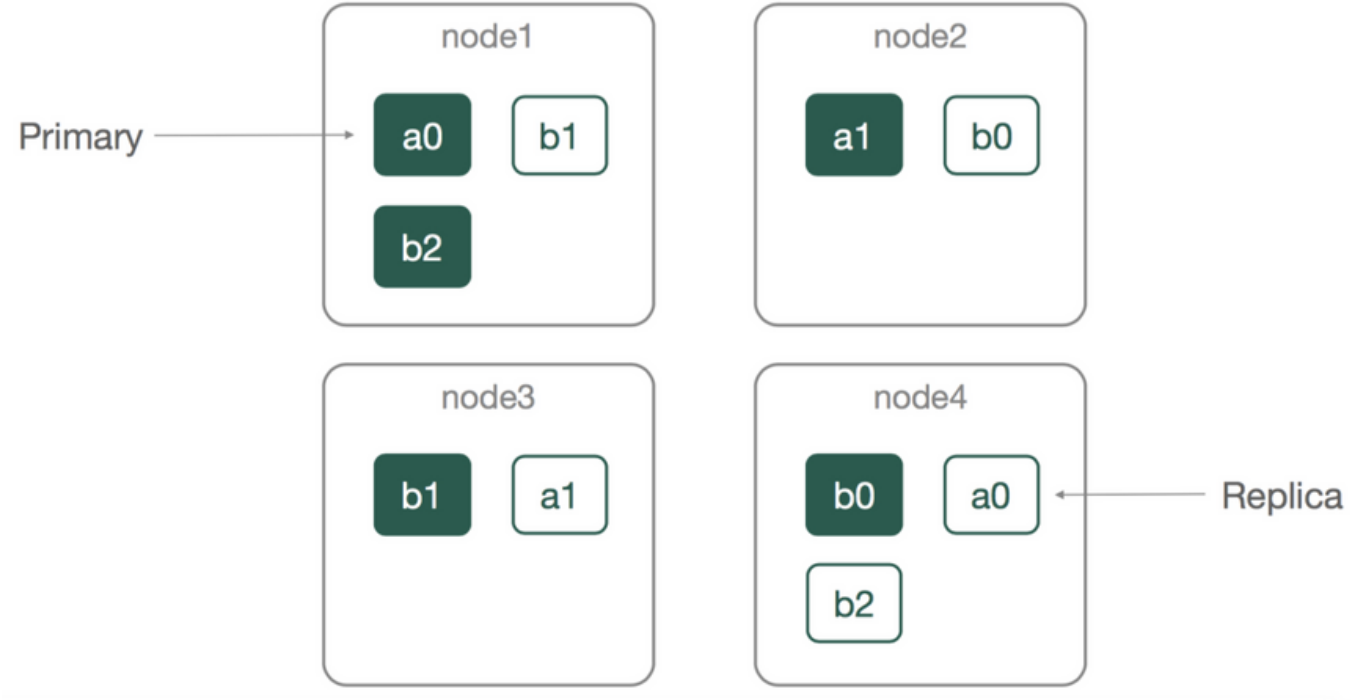
The Elastic stack is obviously useful more much more than full-text product search but we will use only these components for our purpose today:

- Logstash - ingest data (csv, json, ...)
- Elasticsearch - full-text search engine
- Kibana - console and visualization

See more at https://www.elastic.co/what-is/elk-stack

Elastic stack as an architecture

What we will NOT cover today - nodes, shards, replicas

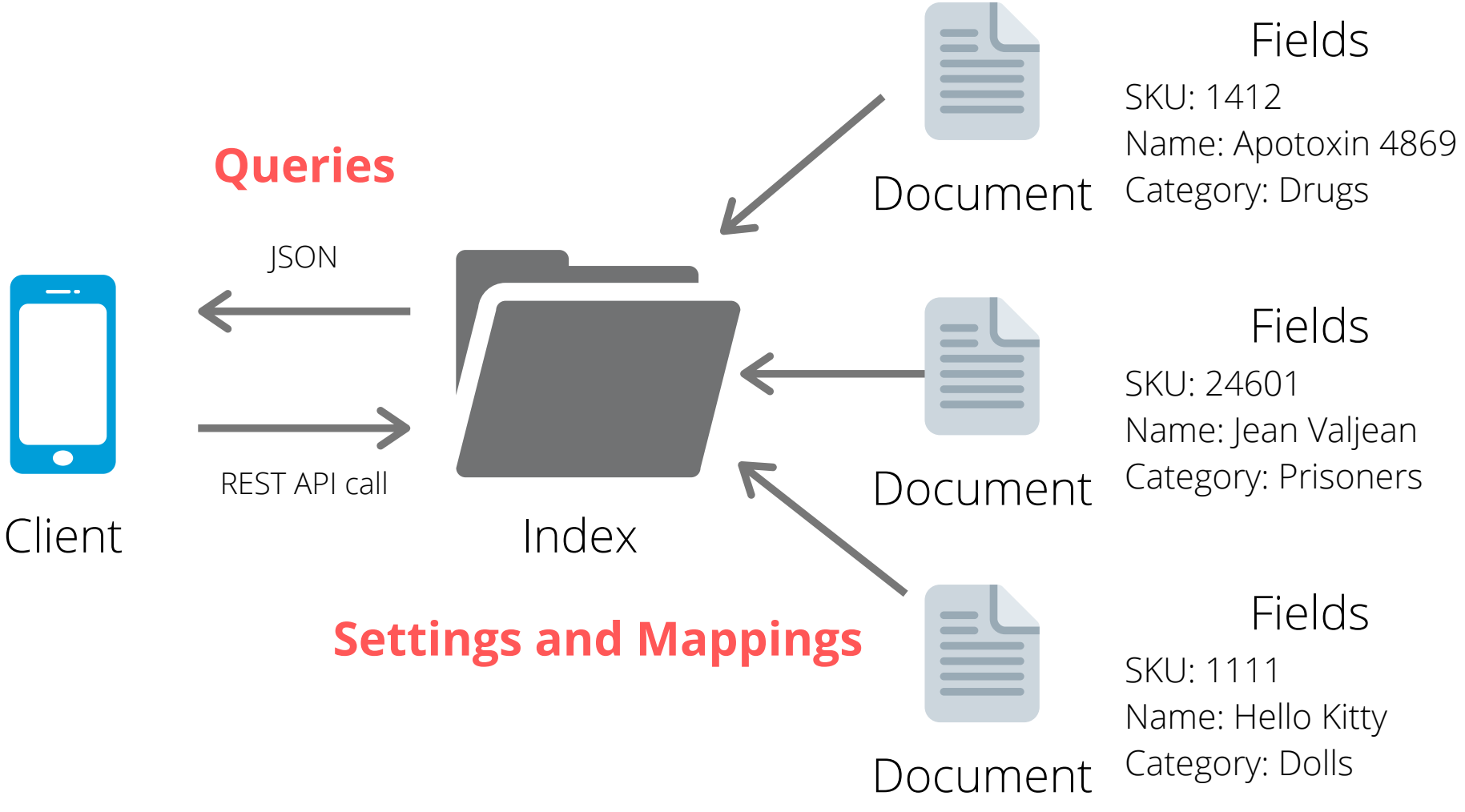


See more at

- https://www.elastic.co/blog/every-shard-deserves-a-home
- https://thoughts.t37.net/designing-the-perfect-elasticsearch-cluster-the-almost-definitive-guide-e614eabc1a87

Elastic stack vocabulary

What you need to know to configure full-text search like a ninja



Part I

Ingest data with logstash

HOW TO TURN CSV/JSON INTO DISTRIBUTED INDICES

Always perform quality check before ingestion

Example from TOPS Supermarket; see tops_sample_qc.ipynb

	name_en	brand_en	category_en	subcategory_en	class_en	long_desc_en
0	Chicken Pie(B	MY CHOICE	Fresh Food & Bakery	Bakery	Danish & Puff Pastry	พายไส้ไก่ หอม อร่อย เ...
1	Heinz Pickled Onions 440g.	HEINZ	Fruit & Vegetables	Preserved Fruit & Ve...	Preserved Vegetables	heinz pickled onions...
2	PC Tuna Salmon Flake 70g.	PC TUNA	Pantry & Ingredients	Canned Food	Instant Meals	nan
3	Malee Pasteurized Mandarin Orange Ju...	MALEE	nan	nan	nan	nan
4	Frontline Plus Kills Fleas And Tick For B...	FRONTLINE	Household & Pet	Pet Care	Pet Health Care	nan
5	ST. Ives Even and Bright Pink Lemon a...	ST.IVES	Health & Beauty Care	Facial Care	Facial Cleanser	nan
6	Sengheng Fresh Tofu Skin 100g.	SENGHENG	Fresh Food & Bakery	Tofu & Fresh Noodle	Tofu	เส่งเฮงฟองเต้าหู้ คุณภ...
7	Brands Essence of Chicken Original 42...	BRANDS	Beverages	Health Tonics	Essence of Chicken	แบรนด์ซุปไก่สกัดรสต้น...
8	Tops Brand Fish Maw 50g.	TOPS	Pantry & Ingredients	Dried Ingredients	Dried Soup Ingredients	nan
9	My Choice Thai Seasoned & Rolled Cut...	MY CHOICE THAI	Meat & Seafood	Marinated Meat	Processed Seafood	มายช้อยส์ไทยปลาหมึ...
10	Cathy Doll Tsum Tsum Oil Control Pact ...	CATHY DOLL	International Products	KOREA	Health & Beauty Care	nan
11	Dr. Hiratake Mushroom 150g.	DOCTOR VEGETAB...	Fruit & Vegetables	Vegetables	Mushroom	ชิตาเกะ หรือเห็ดหอม เ...
12	Healthy Boy Black Soy Sauce 400g.	HEALTHY BOY	Pantry & Ingredients	Sauces	Soy Sauce	nan
13	Tops Spicy Stir Fried Vegetarian Protein...	TOPS	Fresh Food & Bakery	Frozen Food	Frozen Meals	ท็อปส์ข้าวกระเพราเจ ผ...
14	Attack Easy Quick Happy Love Powder...	ATTACK	Household & Pet	Laundry	Powder Detergent	แอทแทคอีซี่ควิกแฮปปี้เ...
15	Boontiang Toffee 200g.	BOONTIANG	Fruit & Vegetables	Preserved Fruit & Ve...	Fruit Desserts	บุญเที่ยงท๊อฟฟี่กะทิใส่...

Import csv the hard way

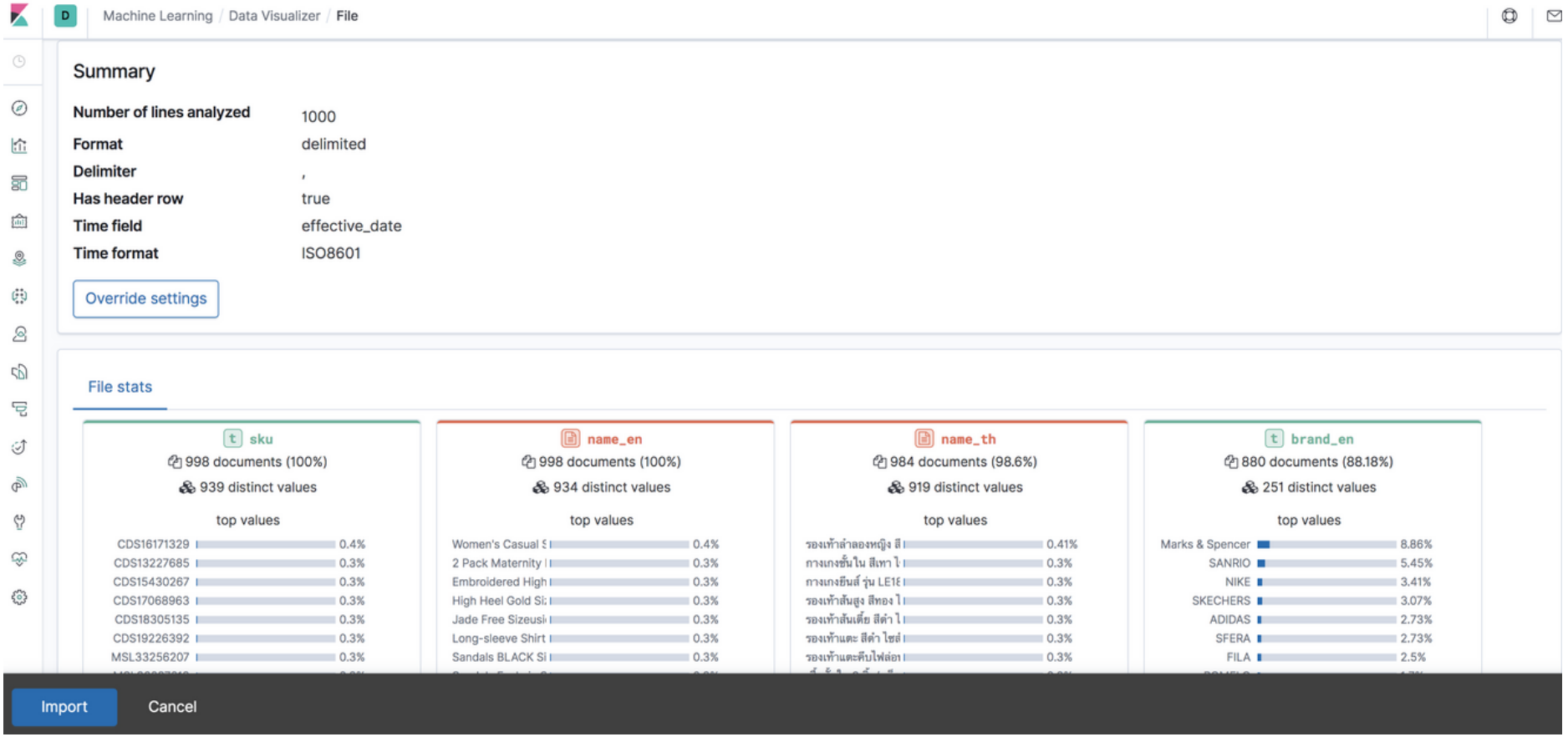
Create config file for logstash; see tops_sample.conf

```
input {
    file {
        path => "/Users/admin/projects/esninja/data/tops_sample.csv"
        start_position => "beginning"
        sincedb_path => "/Users/admin/projects/esninja/data/logs.txt"
    }
}
filter {
    csv {
        separator => ","
        columns => ["sku", "created_at", "name_en", "name_th", "brand_en",
        "brand_th","category_en", "subcategory_en", "class_en",
        "category_th","subcategory_th", "class_th", "long_desc_th", "long_desc_en"]
    }
}
output {
    elasticsearch {
        hosts => ["http://localhost:9200/"]
        index => "tops_sample"
    }
    stdout {}
}
```

See more at https://www.elastic.co/guide/en/logstash/current/configuration.html

Import csv the painless way

Use `ML > data visualizer` on kibana for up to 100 MB



In your local browser: http://localhost:5601/app/ml#/filedatavisualizer

Part II

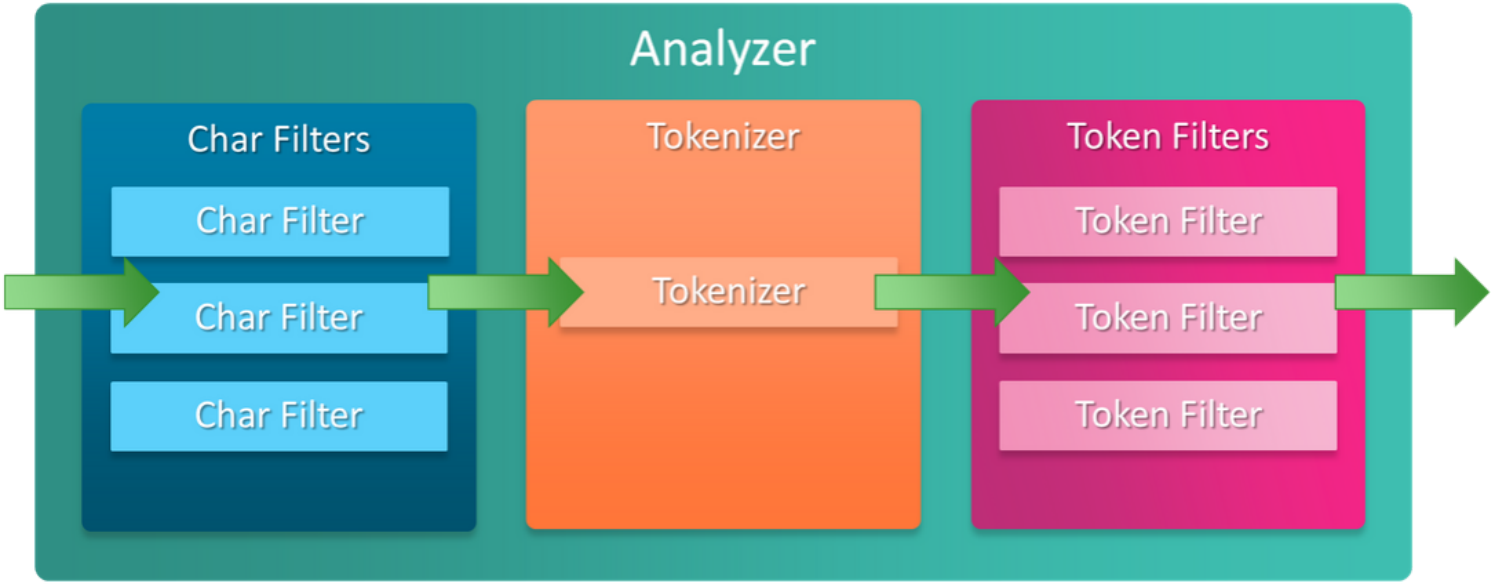
Settings and Mappings

TEACH ELASTICSEARCH HOW TO TREAT YOUR TEXTS

Settings

The tools that elasticsearch can use to process texts

- Analyzer - a text processing function comprise of:
  - Character Filter - process at character level
  - Tokenizer - tokenize characters to tokens
  - (Token) Filter - process at token level



See more at https://www.elastic.co/guide/en/elasticsearch/reference/current/analysis-custom-analyzer.html

Some noteworthy character filters

HTML stripping, connector stripping, number delimiter

```
"char_filter": {
  "connector_flt": {
    "type": "pattern_replace",
    "pattern": "[-'_]",
    "replacement": ""
  },
  "numeric_delim_flt": {
    "type": "pattern_replace",
    "pattern": "([0-9]*\\.?[0-9]+)",
    "replacement": " $1 "
  }
},
```

Personally I only use:

- **`html_strip`** to replace HTML tags

and two custom char filters:

- **connector filter** to remove things like `'` in `l'oreal`
- **numeric delimiter filter** to break things like `pm2.5` to [`pm`,`2.5`]
  * Elasticsearch also has its own `word_delimiter` at token level but I found this works better for my purpose

What tokenizers to use

`thai`, `standard`, `icu_tokenizer`, or none of them

Built-in tokenizers of Elasticsearch

- **standard** - cannot tokenize Thai words at all
- **thai** - pretty bad e.g. `คลิปดำ` (black clips) to [`ค`,`ลิ`,`ป`,`ดำ`] (k, li, p, black)
- **icu_tokenizer** - mostly correct but does not take into account the needs for product search e.g. `หูฟัง` (earphones) to [`หู`,`ฟัง`] (ear, listen) so it will match also [`ต่าง`,`หู`] (earrings tokenized as different, ear)

The solution is to have either

- **Custom dict-based tokenizer plugin in Java** - faster, easier to maintain
- **Tokenize all Thai texts before indexing** with a special character such as pipe - adds another moving part to your system

Analyzer that almost always work

If you have no idea what to do, just start with this

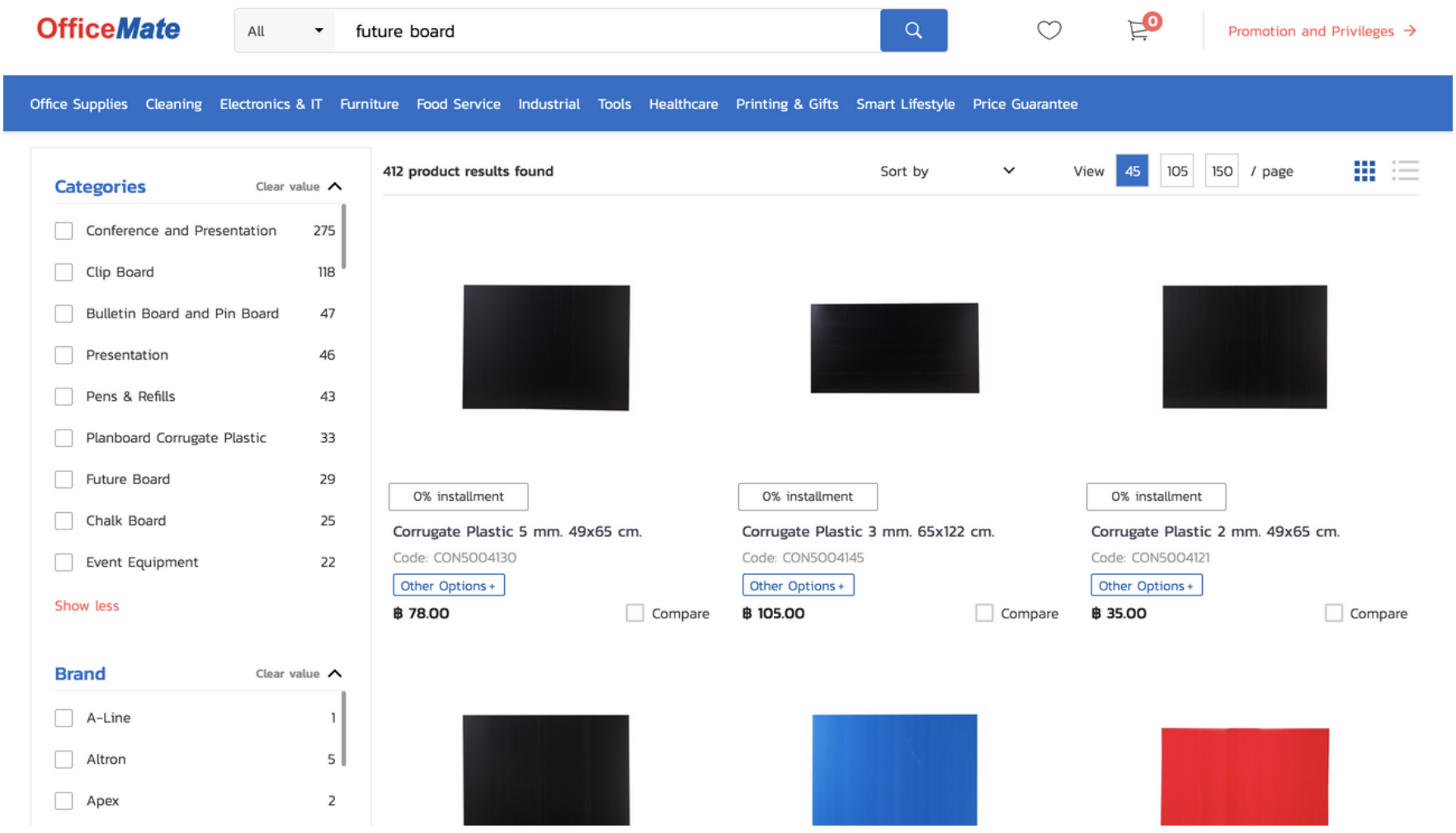
```
"word_stem_anl": {
  "char_filter": [
    "html_strip",
    "connector_flt",
    "numeric_delim_flt"
  ],
  "filter": [
    "lowercase",
    "asciifolding",
    "trim",
    "decimal_digit",
    "snowball_flt",
    "synonym_flt"
  ],
  "tokenizer": "icu_tokenizer"
},
```

A generic analyzer that works for most product search use cases have:

- Tokenization by ICU or better your own custom tokenizer
- `lowercase`
- `asciifolding` to handle things like `Lancôme`
- `trim` to get rid of surrounding spaces
- `decimal_digit` in case someone wants to use Thai numerals
- Snowball stemming to match words at their root forms
- Synonyms

Quick note on the synonym filter

Sometimes the only solution is manual input



Quick note on stemming

You do not always want to match the root forms

With stemming, both `ski` and `skiing` will match both document A and B.

But how do we make sure that document B comes up first when user searches for `skiing`?

Document A

Fields

SKU: 1234
Name: Ski Resort
Category: Travel

Document B

Fields

SKU: 5678
Name: Skiing Equipment
Category: Sports

See more at https://www.elastic.co/guide/en/elasticsearch/reference/6.3/mixing-exact-search-with-stemming.html

Mappings

Instructions on what tools elasticsearch should use to process each text field

```
"name": {
  "type": "text",
  "analyzer": "word_stem_anl",
  "fields": {
    "exact": {
      "type": "text",
      "analyzer": "word_anl"
    },
    "untouched": {
      "type": "keyword"
    }
  }
},
```

We can specify which analyzer to use for each specific text field.

Text fields have two types:

- **text** - full-text values
- **keyword** - exact match only

See more at https://www.elastic.co/guide/en/elasticsearch/reference/6.3/mixing-exact-search-with-stemming.html

Part III

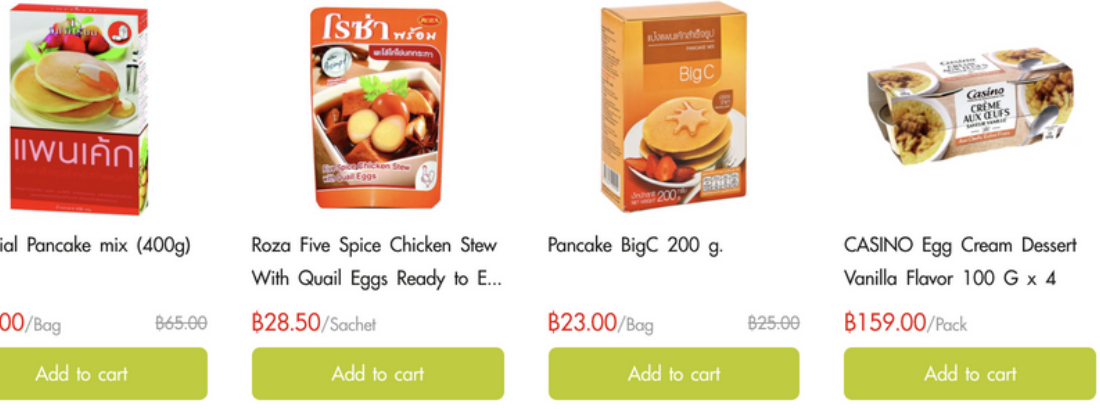
Queries and tests

FIND THE BEST WAY TO QUERY YOUR PRODUCT INDICES AS FAST AS POSSIBLE

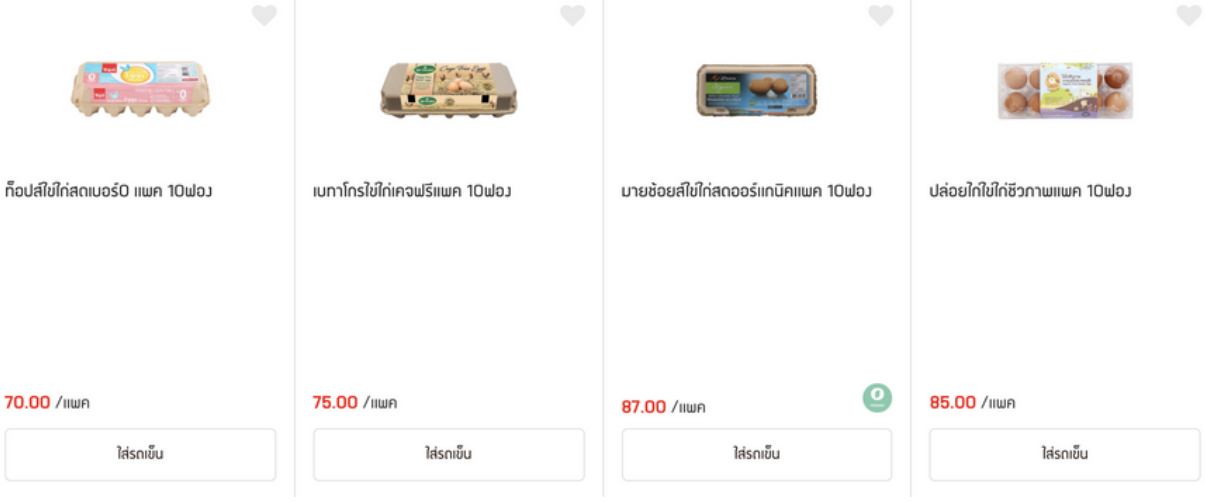
Simple match query

Why it is an extremely bad idea to do simple match query on product name

`eggs` on Big C



`eggs` on Tops



Multi-match query

Your best friend in e-commerce full-text search

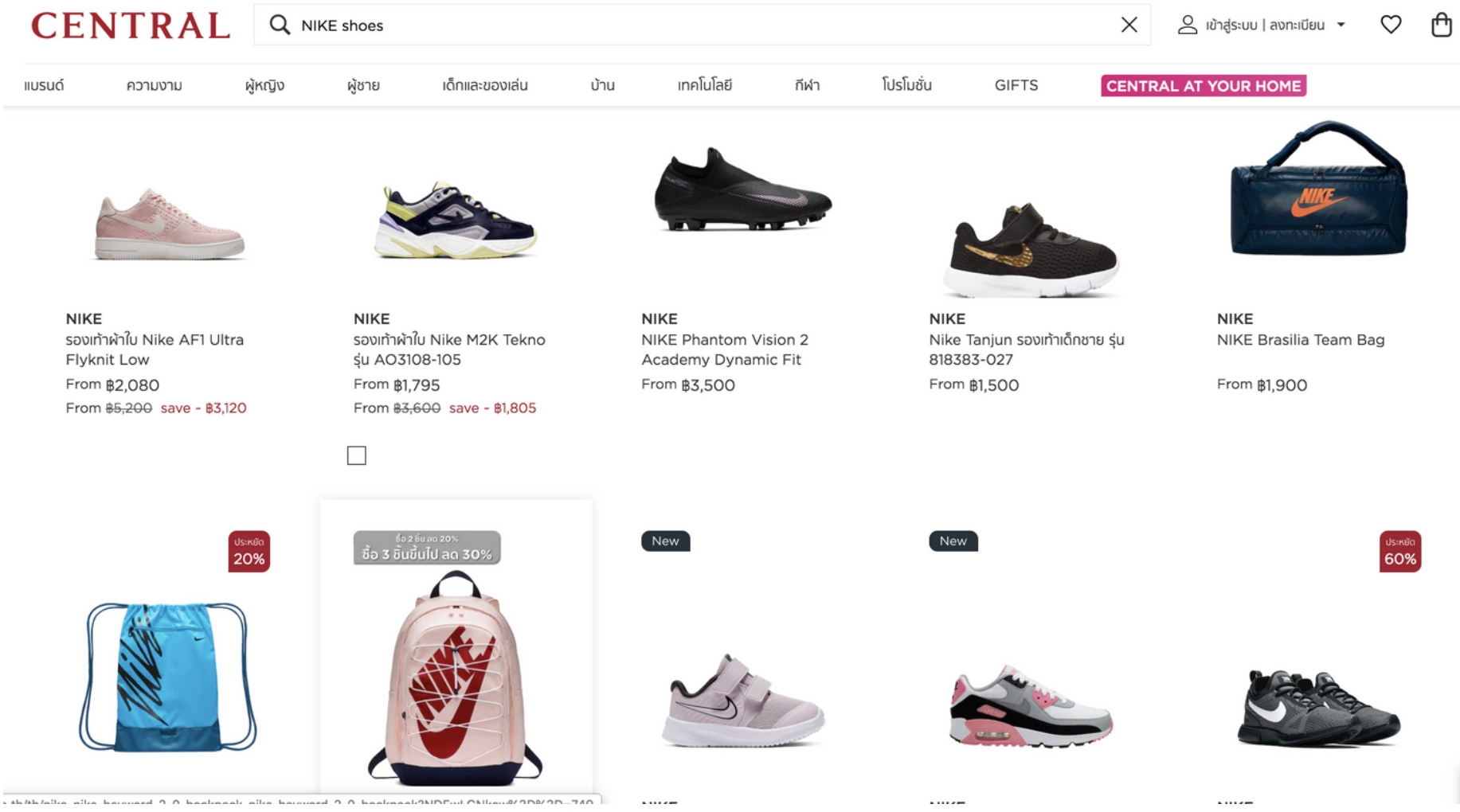
With multi-match queries, you can simultaneously search multiple fields at the same time. I prefer `most_fields` option because it **allows you to give each field a specific weight** and **allows fuzziness**. (not available in `cross_fields`)

`best_fields`	(**default**) Finds documents which match any field, but uses the `_score` from the best field. See `best_fields`.
`most_fields`	Finds documents which match any field and combines the `_score` from each field. See `most_fields`.
`cross_fields`	Treats fields with the same `analyzer` as though they were one big field. Looks for each word in **any** field. See `cross_fields`.

https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-multi-match-query.html

Multi-match query with operator `and`

When you want `nike shoes` to not return `nike shirts`



Combining the queries

boolean query with `should`, `must`, `must_not` and `filter`

Occur	Description
`must`	The clause (query) must appear in matching documents and will contribute to the score.
`filter`	The clause (query) must appear in matching documents. However unlike `must` the score of the query will be ignored. Filter clauses are executed in filter context, meaning that scoring is ignored and clauses are considered for caching.
`should`	The clause (query) should appear in the matching document.
`must_not`	The clause (query) must not appear in the matching documents. Clauses are executed in filter context meaning that scoring is ignored and clauses are considered for caching. Because scoring is ignored, a score of `0` for all documents is returned.

See more at https://www.elastic.co/guide/en/elasticsearch/reference/current/query-dsl-bool-query.html

Boosting

Some fields and boolean queries are more equal than others

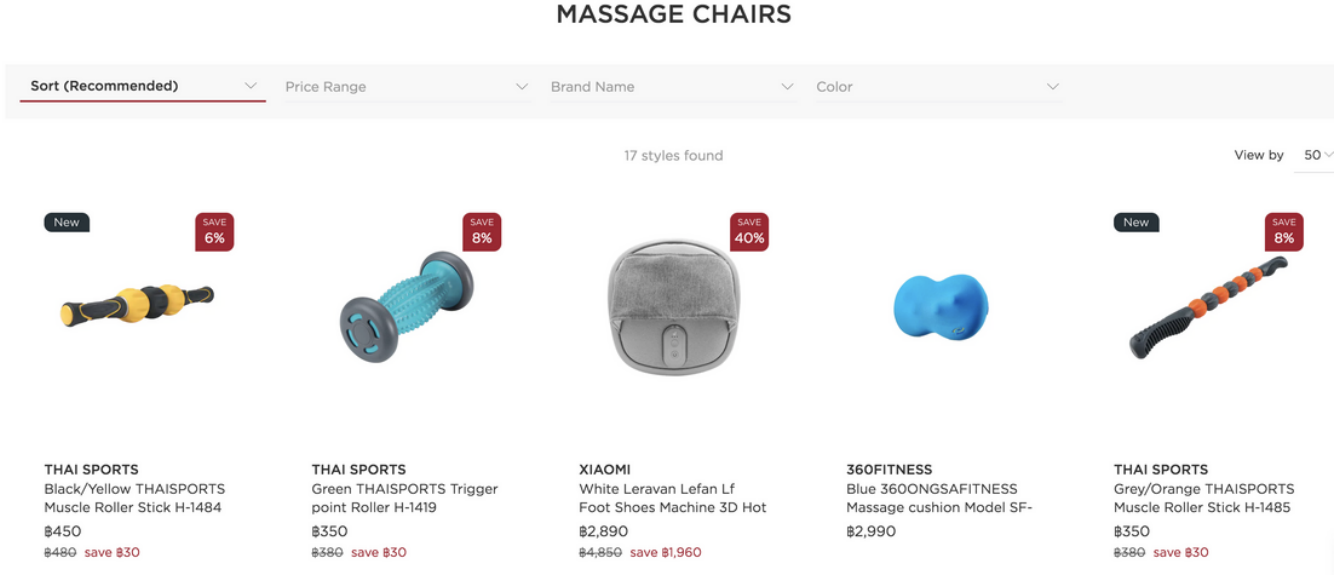
We can combine what we have learned so far with boosting function to allow the search to perform the following hierarchy:

1. Match tokens without stemming
2. Match the entire tokens of the search term
3. Fuzzily match 1. and 2.
4. Go over 1., 2. and 3. but also allows partial match of some tokens of the search term e.g. `eggs` will match `fresh eggs` category

Things to keep in mind

Finetuning elasticsearch mappings and queries is a balancing act

- **Relevance and speed** might be at odds e.g. four multi-matches might take twice the time to run compared to a simple multi-match; use the search profiler
- Generally we prefer **recall over precision** since we will re-rank the results with a model later anyways
- Content team needs to care



Homework

What are some things you could have done better?

- Try creating your own settings, mappings and queries. See if you can pass the test cases in `test_cases.md`
- Is there a better way to test other than doing queries by hand?
- Find a way to solve wrongly tokenized Thai words without using a custom tokenizer.
- Find out why `word_delimiter` is worse than a custom character filter to delimiter numbers and characters. What are the cases where this is not true?