Backend Developer: Take Home Challenge

In this challenge, you will work on three problems. The test tries to target different type of problems we solve every day at Zyft. It should also give you a flavour of what working at Zyft is like.

All the best!!

Please write up and package your solutions as a zip file.

It's OK to use your best judgement if any of the requirements are ambiguous, but state your assumptions if necessary inside a `readme.txt` file.

Submit your answer in a zip file with instructions on how to run it in the `readme.txt` file.

1. HTML to XPath

For this problem, consider a sample block of HTML, e.g.

```
<ul id='favouriteThings' class='myList' zyft_node_id='123456'>
  <li id='item1'> Thing 1</li>
  <li id='item2'> Thing 2</li>
  <li id='item3'> Thing 3</li>
</ul>
```

You need to create a **general** solution to create an `xpath` query from two parameters:

- `zyft_node_id` a string (usually less than 24 characters long).
- The broader `HTML` of a page, a large string (could be html of any webpage, but would have one node with attribute `zyft_node_id`)

For example an answer for the sample block above could be.

```
//li[@id="item1"]/parent::ul
```

Another xpath could be

```
//ul[@id="favouriteThings"]
```

Assume the **zyft_node_id** is an attribute on an element which we need to extract an xpath query for.

Don't assume the id of an element is unique. Also general HTML can be tricky so think about the various scenarios which can happen.

The solution can be written in any language / framework you are familiar with.

More points will be awarded if you can ensure that the query attempts to uniquely select that element (note that you can't use the **zyft_node_id** inside the xpath query).

TLDR:

```
def generate_xpath_query(zyft_node_id, page_html):
    ## Write your code to generate xPath for the element
    # with `zyft_node_id` attribute

# Sample Execution
> generate_xpath_query("123456", "...<ul id='favouriteThings' class='myList' zyft_node_id='123456'>...")
> //ul[@id="favouriteThings"] # Sample output, the answer can be other valid xPath to pick that element uniquely.
```

2. Dataset Creation

For this problem you are sent a sample file: `data_analysis_test_data.csv` your task is to create a new table that showcases all possible matches between products:

A product matches if the `upc` fields are the same.

The expected fields of this resulting dataset are:

1. upc
2. _id1
3. _id2
4. title1
5. title2
6. retailer1
7. retailer2

Note that products that have no matches should be represented in the resulting dataset with null for the fields with the suffix 2 .

Once this is done, please answer the following questions:

- How many products have at least one match in this dataset?
- How many products have no matches?
- how many total matches in this dataset have titles that are different?

For this we would prefer this is written in SQL or Python. You can use any libraries or any flavour of SQL you want as long as you include them in a requirements.txt file and/or note the flavour of SQL you are using.

TLDR:

Task is to create a dataset of matches.

You need to combine the data using UPC as the field to match on. As UPC is a unique identifier for the product. If two products have the same UPC, it essentially means they are the same product.

If you have two products:

218888b41a2a7fa900ecb303	Harry Potter - Harry & Buckbeak Deluxe 1:10 Scale Statue	https://www.ozziecollectables.com/products/harry-potter-harry-buckbeak-deluxe-1-10-scale-statue	ozziecollectables	599	60288
905ec87b3d54ce58e7e1bf59	Tenth Scale Statues--Harry Potter - Harry & Buckbeak Deluxe 1:10 Scale Statue	https://www.globalgear.com.au/brands/tenth-scale-statues-harry-potter-harry-buckbeak-deluxe-110-scale-statue-iko-iro35048.html	globalgear	569	60288

Then you need to create a dataset like:

_id1	_id2	title1	title2	…more fields
218888b41a2a7fa900ecb303	**905ec87b3d54ce58e7e1bf59**	**Harry Potter - Harry & Buckbeak Deluxe 1:10 Scale Statue**	**Tenth Scale Statues--Harry Potter - Harry & Buckbeak Deluxe 1:10 Scale Statue**	…

3. Tag the Model Number in html pages

You are given a list of model numbers such as:

```
PB62WH
CA416
HU6630B
MDC9082-1
CA960
ACL104RA2-10
DBA2182S.
278062
ACL104RA2-3
IEBC001
IEBC001
```

Consider you have a list of such model numbers. The number of such model numbers could be 1 Million.

You are given an HTML, you want to locate if any of these 1 Million Model numbers appear in the product data/text inside Html.

```
class Feature:
  def __init__(self, list_models: List):
    pass
  def tag_html(HTML):
    pass
```

You need to write a class that can be initialised with list of model numbers (approximately 1 million).

Write an efficient function that can tag HTML really quickly.

It's okay to take longer while preprocessing in `__init__`, but `tag_html` functions need to be really efficient.

Guidelines:

1. `__init__` can take a long time even hours
2. `tag_html` should finish tagging a HTML in less than a second.

Output of the function should be start and end indexes of the model number in html and

```
# Outut and sample execution
> feature_tagging = Feature(["HUB6630B", .......1million model numbers])
> feature_tagging.tag_html("<html> <body> ABC HU6630B XYZ </body><html>")
> ["HU6630", {"start_index":18, "end_index": 24}] # Output gives model number, and start, and end index of the model number in html. H in
HU6630 appears at 18 index in html and end of model number B appears at index 24.
```

The solution can be written in any language/framework you are familiar with.