[redacted]

[redacted]

Approach

I have tried to complete this assignment with the assumption that my approach will need to be reproducible with far larger documents. While this task could have been achieved by manually extracting data, I have made use of Python (in a Google Colab environment) and several packages for data extraction and transformation.

Table 1

This table seems to set maximum levels for 'specially controlled substances' when used in particular types of cosmetic products. The same substance may be regulated differently in a different product category.

For example:

Type	Specially-controlled substances	Maximum Lvel/allowed
10. Mouthwash	**formaldehyde**	- not more than 0.1%
	...	
14. Manicure product	**formaldehyde**	- not more than 5.0%

There is a **many-to-many** relationship between substances and types, associated by the "Maximum Lvel/allowed." The entity-relationship diagram based on this table looks like this:

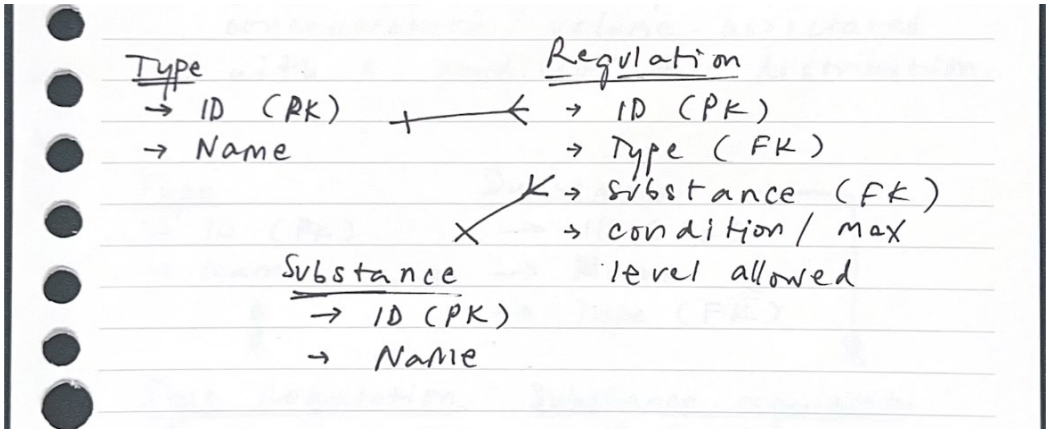


Table 2

This table seems to set out rules for distribution in two different ways:

- "Conditions" and relevant "Notification to a Minister of Public Health" for distribution of **particular categories** of cosmetic products, e.g.:

Type	Controlled substance	Volume Not exceed	condition	Notification of Minister of Public Health
1. Sanitary Napkins	-	-	- Those for external use must made cleanly and hygienically - Those for internal use must be sterilized	(No. 10) B.E. 2536

- "Conditions" and relevant "Notification to a Minister of Public Health" for specially controlled substances, applicable at particular maximum concentrations, e.g.:

Type	Controlled substance	Volume Not exceed	condition	Notification of Minister of Public Health
6. Hair Product that contain anti- dandruff substances	1. Zinc Pyrithione	2.0% 0.5%	Use in rinse-off products Use in leave-on products	(No. 19) B.E. 2537 (No. 19) B.E. 2537

I faced a few challenges with this table, in particular:

Some conditions are identical but may use different wording, e.g. "- Those for external use must made cleanly and hygienically" and "- must be made cleanly and hygienically."

> This would result in duplicate conditions which will either need to be detected when transforming data (e.g. looking for particular words or hard-coding known occurrences), or duplicates may be manually merged.

Some rows specify conditions applicable to different maximum concentrations of the same specially controlled substance.

> E.g.:
>
> ```
> 2. Piroctone Olamine        1.0%   Use in rinse-off products    (No. 19) B.E. 2537
>                             0.1%   Use in rinse-off products    (No. 19) B.E. 2537
> 3. Climbazole               2.0%   Use in rinse-off products    (No. 26) B.E. 2537
>                             0.5%   Use in rinse-off products    (No. 26) B.E. 2537
> ```
>
> With all these values being in a single row, it is difficult to reliably predict when the conditions and maximum concentrations for one substance end, and where the next one starts. However, I have observed some patterns that may help with this.
>
> The most important pattern I noticed is that different specially-controlled substances tend to correspond with different ministerial notifications. E.g., in the table above, "Piroctone Olamine" is regulated by (No. 19) B.E. 2537, whereas Climbazole is regulated by (No. 26) B.E. 2537.
>
> It is true that the same notification applies to Zinc Pyrithione, however, this is recorded in a separate cell, perhaps intentionally.
>
> Another pattern is that there is usually a whole ratio between the number of substances and max. concentrations/conditions/notifications listed. For example, "Products containing UV protection" list **19** substances and **19** max. concentrations (1:1); "Hair product that contains anti-dandruff substances" contains 3 substances and 6 max. concentrations, conditions, and notifications (1:2).
>
> If, once I have extracted the raw data I detect a whole ratio, I could use this to segment the different combinations of maximum concentration, conditions and notification for each substance. This, however, assumes that this ratio does not occur randomly and that these combinations are evenly distributed. For instance, something like this would break such logic:

```
2. Piroctone Olamine         1.0%   Use in rinse-off products    (No. 19) B.E. 2537
                             0.5%   Use in rinse-off products    (No. 19) B.E. 2537
                             0.25%  Use in rinse-off products    (No. 19) B.E. 2537
——*presumed cutoff, based on ratio*————————————————————————————————————
                             0.1%   Use in rinse-off products    (No. 19) B.E. 2537
——*actual cutoff*——————————————————————————————————————————————————————
3. Climbazole                2.0%   Use in rinse-off products    (No. 26) B.E. 2537
                             0.5%   Use in rinse-off products    (No. 26) B.E. 2537
```

As it still has a whole-number ratio, but maximum concentrations are not evenly distributed.

The final pattern that may help differentiate this is that maximum concentrations are always listed in a descending order. By comparing the maximum concentration of the current value with the former value, it can be determined if this now relates to a different substance (if it is greater).

However, this approach also makes some assumptions. Namely that concentrations are **always** listed in descending order; and that the highest concentration of a substance will always be greater than the lowest concentration of the preceding substance. For example:

```
2. Piroctone Olamine         1.0%   Use in rinse-off products    (No. 19) B.E. 2537
                             0.5%   Use in rinse-off products    (No. 19) B.E. 2537
——*missed cutoff*—————————————————————————————————————————————————————
3. Climbazole                0.2%   Use in rinse-off products    (No. 26) B.E. 2537
                             0.1%   Use in rinse-off products    (No. 26) B.E. 2537
```

For the purpose of this assignment, I have taken the first approach of using the ministerial notification as a means of detecting the boundaries for different substances within the row.

If I had been assigned this task at work, I would clear any assumptions with someone else (before selecting an approach) to ensure the output is as useful as possible.

Some conditions have nested bullet points

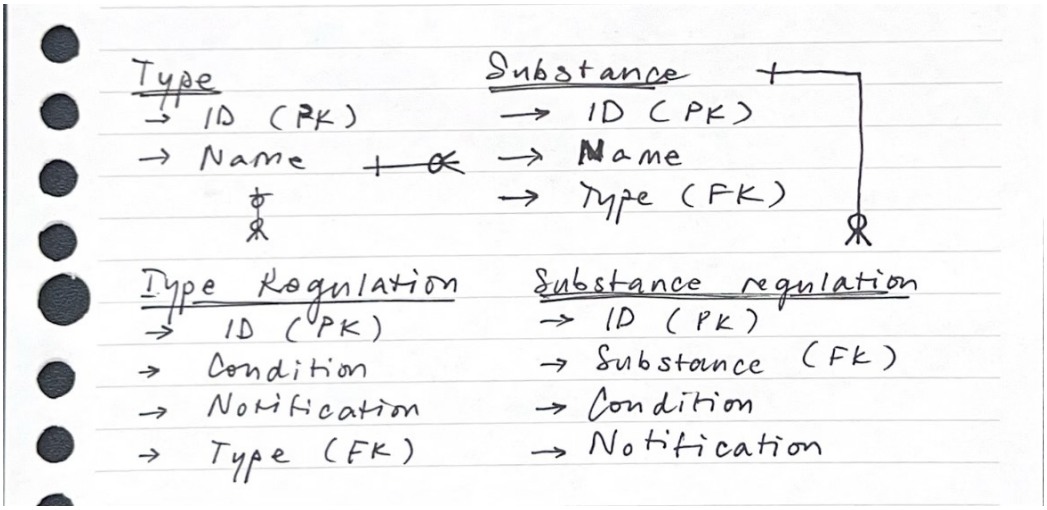
The only occurrences of this are at the final bullet point within a cell which ends with the word “contains.”

Some rows specify regulations both for an entire category (Type) of cosmetic products, as well as for individual substances.

For instance, the row for “Products containing UV protection” specifies maximum concentrations per substance, but imposes a Notification to a Minister of Public Health for the whole product type.

I will record these as separate entities in my final output (i.e., a regulation of a product type and regulations of substances).

The entities from Table 2, and their relationships, can be illustrated like so:



Problems I ran into

When I was trying to transform the Table 1/Listed max. Levels, I ran into an index error— I was trying to access the third cell of the row (the max. level) but it didn't seem to exist.

```
    listed_max_levels = row[2].split('- ')
    print(listed_max_levels)
```

```
['', 'not more than 5.0%\n', 'not more than 0.5%\n', 'not more than 12.0%']
['', 'not more than 11.0% at pH\n7-9.5 calculated in form of\nthioglycolic acid\n', 'not more than 11.0% at
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
<ipython-input-16-7a774b3dc42b> in <cell line: 5>()
     20       ]
     21
---> 22   listed_max_levels = row[2].split('- ')
     23   print(listed_max_levels)

IndexError: list index out of range
```

Next steps: Explain error

Having another look at the JSON data from the extract section, it seems that really is the case:

```
[
  [
    '1. Products for use on\nscalp hair',
    '- oxalic acid including esters and\nalkaline salts of oxalic acid\n- resorcinol\n- hydrogen peroxide, including oth
    '- not more than 5.0%\n- not more than 0.5%\n- not more than 12.0%'
  ],
  [
    '2. Hair Perming\nProduct s',
    '- thioglycolic acid and its salts\n- thioglycolic acid esters\n- thiolactic acid',
    '- not more than 11.0% at pH\n7-9.5 calculated in form of\nthioglycolic acid\n- not more than 11.0% at pH\n6-9.5 cal
  ],
  [
    '3. Hair Straightening Products',
    '- thioglycolic acid and its salts - thioglycolic acid esters\n- potassium hydroxide or sodium\nhydroxide'
  ],    third element doesn't exist
  [
```

This seemed to happen in places where the cell spanned two pages, e.g.:

3. Hair Straightening	- thioglycolic acid and its salts	- not more than 11.0% at pH

Products	- thioglycolic acid esters	7-9.5 calculated in form of thioglycolic acid - not more than 11.0% at pH 6-9.5 calculated in form of thioglycolic acid
	- potassium hydroxide or sodium hydroxide	- not more than 2.0% calculated in form of sodium hydroxide

After revising the extraction code, the culprit was this 2:

```
elif len(table1_data) > 1 and not begins_with_number:
    prev_row = table1_data[-1]
    new_row = [prev_row[i] + " " + row[i] for i in range(2)]
    print(new_row)
    table1_data[-1] = new_row
```

This is the line which combines the contents of cells which overflow between pages. It uses list comprehension to combine the cells in each column. There are 3 columns, however, thinking about 0-based indices (i.e. [0, 1, 2]), I mistakenly entered 2 for the number of columns, instead of 3. The third column was being skipped in all cells which overspilled:

```
['3. Hair Straightening Products', '– thioglycolic acid and its salts – thioglycolic acid esters\n–
potassium hydroxide or sodium\nhydroxide']
['7. Hair Bleaching\nProducts ', '– persulfates of ammonium or\npotassium or sodium ']
['14. Manicure Product ', '– formaldehyde\n– hydrogen peroxide including other\nperoxide except
sodium peroxide\n– potassium hydroxide or sodium\nhydroxide used as solvent in nail products']
```

. . .

Once I had successfully extracted all substances and maximum levels, I ran a quick check to see if their number matched up in every row.

```
    type_name = re.sub(type_number_regex, '', type_name_clear_newline)

    # Cleaning up the names of substances, ignoring
    # first element from the split, as Python outputs an array as if
    # there is an empty element before the "–" delimiter
    listed_substances = row[1].split('– ')
    listed_substances = [
        # Removing trailing spaces and replace newlines with spaces
        substance.strip().replace('\n', ' ')
        for substance
        in listed_substances[1:]
        ]

    listed_max_levels = row[2].split('– ')
    listed_max_levels = [level.strip() for level in listed_max_levels[1:]]

    # checking that there are no discrepancies
    print(len(listed_substances) == len(listed_max_levels))
```

```
True
True
True
False      oh no
True
True
True
True
True
True
True
True
True
True
True
True
True
True
```

Some quick debugging revealed an unexpected single-letter element in the substances list (‘m’)

```
    if len(listed_substances) != len(listed_max_levels):
      print(listed_substances)
      print(listed_max_levels)
```

```
-substituted derivatives and their salts (except 4-methyl-m- phenylenediamine and its salts)', 'm', 'or p-phenylenediamines
ot more than 6.0%', 'not more than 5.0%']
```

This corresponds with “4-methyl-m-phenylenediamine” which wraps onto a new line.

4. Hair dyeing Products	- diaminophenols	- not more than 10.0%
	- hydroquinone	- not more than 2.0%
	- methylphenylenediamines including N-substituted derivatives and their salts (except 4-methyl-m-phenylenediamine and its salts)	- not more than 10.0%
	- m- or p-phenylenediamines including N-substituted derivatives	- not more than 6.0%

To remediate this, I changed the delimiter on the split expression to be “\n- “ to factor in a new line prior to the dash denoting a bullet point. This also removes the need to omit the first empty element in the array.

This, however, caused another issue: the bullet points “'- thioglycolic acid and its salts - thioglycolic acid esters\n- potassium hydroxide or sodium\nhydroxide’” were not separated into different elements.

This arose from the way that pages are concatenated. I amended concatenation of cells between pages to add a line-break to fix this.

Counts of substances and maximum levels now match for all rows.

. . .

I faced a bit of a challenge with assigning primary keys and ensuring no duplicates when transforming the data from the first table.

I was initially planning on having three pandas DataFrames, but it seems that this would require a lot of manual checks to properly upsert substances while making sure there are no duplicates and correctly referencing primary keys.

Instead, I am going to try sqlite3, as it’s suited to handling relational data. At the end, I could simply query the database and put the records in a DataFrame to still make use of the Excel export functionality offered by pandas.

This approach worked out as planned:

Name	Type	Schema
Tables (3)		
Regulations		CREATE TABLE Regulations (ID INTEGER PRIMARY KEY, TypeID INTEGER, SubstanceID INTEGER, MaxLevelRegulatoryWording TEXT, FOREIGN KEY (T
Substances		CREATE TABLE Substances (ID INTEGER PRIMARY KEY, SubstanceName TEXT UNIQUE)
Types		CREATE TABLE Types (ID INTEGER PRIMARY KEY, TypeName TEXT UNIQUE)

Table: Substances

	ID	SubstanceName
	F...	Filter
1	1	– oxalic acid including esters and alkaline …
2	2	resorcinol
3	3	hydrogen peroxide, including other peroxides …
4	4	– thioglycolic acid and its salts
5	5	thioglycolic acid esters
6	6	thiolactic acid
7	7	potassium hydroxide or sodium hydroxide
8	8	– diaminophenols
9	9	hydroquinone
10	10	methylphenylenediamines including N–…
11	11	m– or p–phenylenediamines including N–…
12	12	– lead acetate
13	13	silver nitrate
14	14	– aluminium pyrithione
15	15	– persulfates of ammonium or potassium or …
16	16	– chlorates of the alkali metals
17	17	salts and derivatives of fluoride
18	18	– salts and derivatives of fluoride
19	19	– formaldehyde
20	20	hydrogen peroxide including other peroxide …
21	21	cetylpyridinium chloride
22	22	– cetylpyridinium chloride
23	23	– calcium sulfide
24	24	calcium thioglycolate
25	25	strontium sulfide
26	26	thioglycolic acid and its salts
27	27	zinc sulfide
28	28	– hydrogen peroxide including other peroxide …
29	29	hydrogen peroxide including other peroxide …
30	30	potassium hydroxide or sodium hydroxide used …
31	31	– zinc p–phenolsulfonate
32	32	– 1,1,1–trichloroethane or solvent …
33	33	– chorates of alkali metals

Table: Types

	ID	TypeName
	F...	Filter
1	1	Products for use on scalp hair
2	2	Hair Perming Product s
3	3	Hair Straightening Products
4	4	Hair dyeing Products
5	5	Hair Blackening Cream
6	6	Anti– dandruff products
7	7	Hair Bleaching Products
8	8	Toothpaste
9	9	Dental Floss
10	10	Mouth wash
11	11	Oral Refresher Spray
12	12	Deplilatory products
13	13	Skin Hair Lightening Products
14	14	Manicure Product
15	15	Deodorant products
16	16	Spray products
17	17	Other products besides toothpaste

Table: Regulations

	ID	TypeID	SubstanceID	MaxLevelRegulatoryWording
	F...	Filter	Filter	Filter
1	1	1	1	– not more than 5.0%
2	2	1	2	not more than 0.5%
3	3	1	3	not more than 12.0%
4	4	2	4	– not more than 11.0% at pH 7–9.5 calculated …
5	5	2	5	not more than 11.0% at pH 6–9.5 calculated in…
6	6	2	6	not more than 8.5% at pH not more than 9.5
7	7	3	3	– not more than 11.0% at pH 7–9.5 calculated …
8	8	3	7	not more than 11.0% at pH 6–9.5 calculated in…
9	9	3	7	not more than 2.0% calculated in form of …
10	10	4	8	– not more than 10.0%
11	11	4	9	not more than 2.0%
12	12	4	10	not more than 10.0%
13	13	4	11	not more than 6.0%
14	14	4	13	not more than 5.0%
15	15	5	12	– Calculated in the form of lead not more tha…
16	16	5	13	not more than 5.0%
17	17	6	14	– not more than 2.0%
18	18	7	15	– not more than 45% calculated in the form of…
19	19	8	16	– not more than 5.0%
20	20	8	17	not more than 0.11% (active fluoride ion not …
21	21	9	18	– not more than 0.11% (active fluoride ion no…
22	22	10	19	– not more than 0.1%
23	23	10	20	not more than 0.5%
24	24	10	23	not more than 0.11% (active fluoride ion not …
25	25	10	21	not more than 0.06%
26	26	11	22	– not more than 0.06%
27	27	12	23	– not more than 30.0%
28	28	12	24	not more than 9.0% at pH 9– 12.5
29	29	12	25	not more than 20.0%
30	30	12	26	not more than 5.0 at pH 7– 12.7 (Calculate in…
31	31	12	27	not more than 40.0%
32	32	13	28	– not more than 4.0%
33	33	14	14	– not more than 5.0%
34	34	14	29	not more than 2.0%
35	35	14	30	not more than 5.0% calculate in form of sodiu…

1 - 35 of 38

Finally, I queried the data from the SQLite DB, put it into pandas DataFrames, and exported into an Excel spreadsheet.

. . .

Moving onto the second table, I'm using a similar approach of extracting substances, max. volume allowed, conditions, and notifications. I ran a quick check to see if everything matches up in rows where there are listed specially controlled substances.

```
    print(len(listed_substances), len(max_volumes), len(conditions), len(notifications))
```

```
19 19 1 1
1 2 1 2
2 4 1 4
```

Things seemed to check out for the "Products containing UV protection" type, where 19 substances are listed, as well as 19 maximum volume %s.

However, when it came to the last two rows, relating to the "Hair product that contain anti-dandruff substances," there was a slight mismatch— only a single condition was being detected.

6. Hair Product that contain anti-dandruff substances	1. Zinc Pyrithione	2.0% 0.5%	Use in rinse-off products Use in leave-on products	(No. 19) B.E. 2537 (No. 19) B.E. 2537
	2. Piroctone Olamine	1.0% 0.1%	Use in rinse-off products Use in leave-on products	(No. 19) B.E. 2537 (No. 19) B.E. 2537
	3. Climbazole	2.0% 0.5%	Use in rinse-off products Use in leave-on products	(No. 26) B.E. 2537 (No. 26) B.E. 2537

Upon closer inspection, this is due to the fact that conditions in these rows are not listed as bullet points, but simply as sentences. As I am using a RegEx which detects a line-feed (\n) followed by a dash, and uses it as a delimiter to split the string, these cells end up not being split at all.

Since conditions also include line-breaks, I'm not sure what would be a straightforward way of differentiating where one condition begins and ends. I could make inferences (similar to the way I explained earlier about matching up conditions with substances), however, I would need to make a lot of assumptions, which may not translate to what a larger document may look like.

I attempted to use a different table extraction library, PyMuPDF, as it provides greater control.

```
tabs = page.find_tables()
for row in tabs[0].rows:
  for cell in row.cells:
    print("-------")
    bounding_box = cell
    blocks = page.get_text_blocks(clip=bounding_box)

    for block in blocks:
      print(block[4:][0].split("\n"))
```

```
-------
['6. Hair ', 'Product that ', 'contain anti-', 'dandruff ', 'substances  ', '']
-------
['1. Zinc Pyrithione ', '']
-------
['2.0% ', '']
[' ', '']
['0.5% ', '']
-------
['Use in rinse-off ', 'products ', 'Use in leave-on ', 'products ', ' ', '']
-------
['(No. 19) ', '']
['B.E. 2537 ', '']
['(No. 19) ', '']
['B.E. 2537 ', '']
-------
[' ', '']
-------
['2. Piroctone Olamine ', ' ', ' ', ' ', '3. Climbazole ', '']
-------
['1.0% ', '']
[' ', '']
```

I could now get a better idea of the lines of each cell and match-up the values in the different columns.

Splitting the cells into lines now lets me try to pad values downwards to extract the subrows:

```python
# Function which gets the lines of text (as they appear visually) in a cell
def get_lines(cell):
  blocks = get_blocks_in_cell(cell)
  lines_acc = []
  for block in blocks:
    lines_acc.extend(block[4:][0].split("\n"))
  return lines_acc

def split_row_into_padded_subrows(row, ignore_notification=False):
  subrows = []
  substance_lines = get_lines(row.cells[1])

  current_substance = ""

  for line in substance_lines:
    # If the current line begins with digits, followed by a dot, it is a
    # new substance
    pattern = r"\d+\.\s*"
    if re.match(pattern, line):
      current_substance = re.sub(pattern, "", line)
    print (current_substance)



doc = pymupdf.open('/content/Thailand cosmetics guidelines.pdf')
page = doc[8]
tabs = page.find_tables()
for row in tabs[0].rows:
  split_row_into_padded_subrows(row)
```

```
Zinc Pyrithione
Zinc Pyrithione
Piroctone Olamine
Piroctone Olamine
Piroctone Olamine
Piroctone Olamine
Climbazole
Climbazole
```

```python
  current_substance = ""
  current_max_volume = ""
  current_condition = ""
  current_notification = ""

  for i in range(len(substance_lines)):
    # If the current line begins with digits, followed by a dot, it is a
    # new substance
    pattern = r"\d+\.\s*"
    if re.match(pattern, substance_lines[i]):
      current_substance = re.sub(pattern, "", substance_lines[i])
    if max_volume_lines[i] != "":
      current_max_volume = max_volume_lines[i]
    if condition_lines[i] != "":
      current_condition = condition_lines[i]

    print(current_substance, current_max_volume, current_condition)



doc = pymupdf.open('/content/Thailand cosmetics guidelines.pdf')
page = doc[8]
tabs = page.find_tables()
for row in tabs[0].rows:
  split_row_into_padded_subrows(row)
```

```
Zinc Pyrithione  2.0%  Use in rinse-off
Zinc Pyrithione  2.0%  products
Piroctone Olamine  1.0%  Use in rinse-off
Piroctone Olamine  1.0%  products
Piroctone Olamine    Use in leave-on
Piroctone Olamine    products
Climbazole  0.1%  Use in rinse-off
Climbazole  0.1%  products
```

• • •

[redacted]