Bui Nguyen Kim Hai
htlulem185@gmail.com

System Integration Backend Preliminary Test Case: Create product feed from database

1 Task

A product feed is an essential medium for exporting and syncing the product catalog to external systems such as marketplaces and B2B partners. Your objective in this task is to create an XML product feed from a given database of products according to the Google Merchant product data specifications. Some required fields have been omitted from this task for simplicity.

Your implementation must generate a file named 'feed.xml', including the following fields:

- ID [id]
- Title [title]
- Description [description]
- Link [link]
- Image link [image_link]
- Additional image link [additional_image_link]
- Availability [availability]
- Price [price]
- Brand [brand]
- Condition [condition]

Additional details:

- The field values included in the product feed must conform to Google Merchant specifications
- Disabled products (with status 0) must not be included in the feed
- All prices are in Hungarian Forints (HUF)
- Brand represents the product manufacturer
- All products are sold as new
- The base domain for product image URLs is butopea.com, for example: https://butopea.com/image/catalog/DEJEL-NS3TE[D]_1.jpg
- Additional images must be loaded in their respective sort orders
- The product link can be constructed by appending the product ID to https://butopea.com/p/, for example: https://butopea.com/p/3927
- The product catalog data is typically stored in a relational database. To avoid setting up a database server, you will be using a file-based database known as SQLite.

2 Plans

Since this is a blind task, I need to get to know about the database, after using investigating, the database has 4 tables: **manufacturer**, **product**, **product_description**, and **product_image**.

In addition, I need to know about the columns in each table too. For the sake of convenience, i will hight light what column name I will use in each table too.

- **manufacturer**:
  - manufacturer_id
  - name
- **product**:
  - product_id
  - model
  - ean
  - quantity
  - image
  - manufacturer_id
  - price
  - status
- **product_description**:
  - product_id
  - name
  - description
- **product_image**:
  - product_image_id
  - product_id
  - image
  - sort_order

After that, I need to map each column name above to a expected field name.

- ID [id] $\longrightarrow$ product.product_id
- Title [title] $\longrightarrow$ product_description.name
- Description [description] $\longrightarrow$ product_description.description
- Link [link] $\longrightarrow$ "https://butopea.com/p/" + product.product_id
- Image link [image_link] $\longrightarrow$ "https://butopea.com/" + product_image.image
- Additional image link [additional_image_link] $\longrightarrow$ (**product has more than 1 image**) && (select all images that are not the chosen image)
- Availability [availability] $\longrightarrow$ product.quantity > 0 ? 'in_stock' : 'out_of_stock'
- Price [price] $\longrightarrow$ product.price HUF
- Brand [brand] $\longrightarrow$ manufacturer.name
- Condition [condition] $\longrightarrow$ 'new'

There are additional requirements from Google Merchant also, some of the things I think I need to check are:

- product.status $\neq 0$
- product.price $\neq 0$
- product.product_id.length ≤ 50
- product_description.name.length ≤ 150
- product_description.description.length ≤ 5000
- manufacturer.name.length ≤ 70

After then I can write code and turn it to XML using Python libs.

3 Obstacle

The problem comes when joining tables, one product can have more than 1 image, so this will cause duplicated products when we convert it into xml file

Example:

- product1 $\longrightarrow$ image_link: 1, additional_image_link: [2, 3, 4]
- product1 $\longrightarrow$ image_link: 2, additional_image_link: [1, 3, 4]
- product1 $\longrightarrow$ image_link: 3, additional_image_link: [1, 2, 4]
- product1 $\longrightarrow$ image_link: 4, additional_image_link: [1, 2, 3]

I resolve this problem in 2 step:

1. Map the product to the "first image" (i.e. the image has the smallest **sort_order**). It is going to be the sub-query of my main query, where I join tables (included the sub-query too) with some constraints to get the desired output.
2. After retrieving the products, foreach product then I will find the list of additional image link (if any) by using only **product_image** table with given product_id and sort ascending by sort_order, limit 10 rows, offset the first one (because it is the image_link already!).

4 Code

- **data.sqlite:** The database.
- **plans.txt:** The first draft of this document.
- **db.py:** Using **sqlite3** to connect to the db, **fetch** function allows user to fetch data from the db with given query. Results are return in the form of a Python dictionary.
- **my_mxl.py:** Using **xml.etree.ElementTree** draw the XML file, **xml.dom.minidom** is used to print XML file in a pretty format (human-readable). **gen** function allows user to save a list of products into a XML file that follows the RSS 2.0 specification. User may choose to have a readable XML also.
- **main.py:** Run the program, implement the desired queries, use **db.py** and **my_xml.py**.
- **feed.xml:** The result.

5 How to run

I believe you know, but just in case :D non-tech guys read this docs somehow.

1. Pull my github repo: git clone https://github.com/whynotkimhari/butopea.part.1
2. Open the project, ensure you are in the project folder: %YOUR_PATH%/butopea.part.1/
3. Run the following command: python ./main.py ./data.sqlite ./feed.xml true
4. You will not see any response if the program works correctly, the feed.xml will appear in the current folder instead.

——————the end——————