SQL Project (Fabio Costantino)

My role in the Company

As a Content Analyst in the Localization Team, I am responsible for migrating data in and out of our Content Management System called STEP. These are mostly text data - a.k.a. content, which are then published daily in our website for all customers to consume.

As a Localization team, we mainly deal with translated text in 12 different languages, therefore also managing the flow of translation files in and out of the company. In order to translate our content, I have daily contact with external translation agencies and manage the use of a Translation Management System called XTM, which helps organize all files and workflows in a single tool.

Besides, we also provide our manager(s) and the rest of the team with reports of many kind: from tailored collection of data coming from different sources, to reports on products (or families of products) performance.

Some of the tools I use on a daily/weekly basis to perform my job are: **STEP Content Management System**, **XTM Translation Management System**, **Microsoft SQL Server Management Studio**, **Report Builder** from **Adobe Analytics**, **DOMO** as a BI dashboard tool, **Office package**, **Jupyter Notebook** (with **Python**).

Brief

Produce a report for the Japanese market that identifies the Primary Keywords used in the website for all live products

Business problem

The Primary Keywords - **PKW** for short - are very important for SEO performance, since they appear in page titles and in the content to show web crawlers what a particular page is about. For this reason, our Content Enrichment Executives (or CEEs, responsible for enriching the content for the website) are often asked to review the PKWs of entire families of products to ensure the best quality.

Due to the taxonomical structure of our website, that at the time of writing divides the content into 27 categories, PKWs are scattered around our system and the Content Management System produces weekly 27 spreadsheets containing them all. We need to periodically collate all this information to produce reports relevant to our business.

Current process

The current process involves the following steps: importing each of the 27 files into a single text file; ensure that the encoding is correctly setup for visualizing Asian languages and special characters; rename each text file correctly not to be mixed up with other sections of the website; importing each of the 27 text files into various tables in SQL; query the system to obtain information.

Current issue

Considering the amount of PKWs involved in each file, the migration of data is extremely slow and the whole process might take up to 2-3 hours a week - and this without considering Excel file crashing or the chance of human error in such a manual job.

Proposed solution

In order to answer my brief, and similar ones in the future, I decided to do the following:

Step 1 - Create a script in Python that collates all the PKWs with a few clicks.

STEP 2 - Create a new table in SQL that contains all PKWs. I will do this for all languages in one table, so that I can reuse in the future the same table to produce similar reports. This can be efficiently used by all members of the team.

STEP 3 - The last part of my project involves all the queries I need to produce the final report to present to my Line Manager as per brief.

Pros and Cons

Creating this script will allow me to reduce the time spent completing similar tasks from around 3 hours to roughly 15 minutes, which gives me more time to concentrate on other, less mechanical, tasks.

I foresee no detrimental effect in my approach at the time of writing.

0. Importing libraries

First of all I need to import all relevant Python libraries.

In [1]:
```
# Pandas is imported for most of the manipulation of data and information in t
he dataset.
import pandas as pd

# Importing the modules necessary to read/write .csv and .xlsx files.
import csv
import xlsxwriter
import fileinput

# Importing the modules necessary to deal with multiple files with the same ex
tension.
import glob
import os

# Importing the library necessary to display screenshots in Jupyter notebook.
from IPython.display import Image
```

1 Creating a master file

1.1 Importing the Primary Keywords files

The first set of data that I need to retrieve from our Content Management System is the complete list of PKWs used in the website, divided into 27 files, since it reflects the structure of our website divided into 27 sections.

In order to use these files in SQL and import them efficiently, I need to create a single `.csv` file that contains all the information. This is achieved with a function called `concatenate()` that will append each file into a master file called **PKW_master.csv**.

In [2]:
```
# This code prepares a function to concatenate all files into one master .csv
 file.
# The attribute "encoding" allows us to read/write correctly special character
s and Asian languages.

def concatenate(indir='[redacted]
[redacted]27_files_here',
                outfile='[redacted]
[redacted]PKW_master.csv'):
    os.chdir(indir)
    fileList=glob.glob('*.txt')
    dfList=[]
    for filename in fileList:
        print(filename)
        df=pd.read_csv(filename, sep='\t', header=None, low_memory=False)
        dfList.append(df)
    concatDf=pd.concat(dfList, axis=0)
    concatDf.to_csv(outfile, encoding='utf-8', sep='\t', index=None)
```

In [3]:
```
# This code does the actual concatenation and creates the "PKW_master.csv" fil
e in the same folder of this notebook.
# In the previous code, the line "print(filename)" gives the possibility to vi
sualize when each file has been
# worked on, creating a list for us as the function progresses.

concatenate()
```

```
Primary Keyword_SC_ACG_19-09-19_04-16-55.txt
Primary Keyword_SC_AM_19-09-19_04-17-04.txt
Primary Keyword_SC_AST_19-09-19_04-17-00.txt
Primary Keyword_SC_Batt_19-09-19_04-15-20.txt
Primary Keyword_SC_CO_19-09-19_04-03-55.txt
Primary Keyword_SC_CP_19-09-19_04-14-29.txt
Primary Keyword_SC_CW_19-09-19_04-15-17.txt
Primary Keyword_SC_DO_19-09-19_04-04-30.txt
Primary Keyword_SC_EN_19-09-19_04-01-29.txt
Primary Keyword_SC_FM_19-09-19_04-01-35.txt
Primary Keyword_SC_FS_19-09-19_04-00-55.txt
Primary Keyword_SC_FT_19-09-19_04-00-37.txt
Primary Keyword_SC_HV_19-09-19_04-00-11.txt
Primary Keyword_SC_LT_19-09-19_04-00-28.txt
Primary Keyword_SC_OS_19-09-19_04-06-31.txt
Primary Keyword_SC_PCB_19-09-19_04-06-18.txt
Primary Keyword_SC_PC_19-09-19_04-09-20.txt
Primary Keyword_SC_PH_19-09-19_04-05-35.txt
Primary Keyword_SC_PP_19-09-19_04-06-26.txt
Primary Keyword_SC_PT_19-09-19_04-06-14.txt
Primary Keyword_SC_RB_19-09-19_04-09-23.txt
Primary Keyword_SC_RL_19-09-19_04-04-45.txt
Primary Keyword_SC_SM_19-09-19_04-12-56.txt
Primary Keyword_SC_SS_19-09-19_04-05-03.txt
Primary Keyword_SC_SW_19-09-19_04-09-45.txt
Primary Keyword_SC_TM_19-09-19_04-14-13.txt
Primary Keyword_SC_TO_19-09-19_04-13-35.txt
```

In [4]:
```
# This code reads the .csv file, so that I can work on it in the Python enviro
nment.

PKW_master = pd.read_csv('[redacted]
[redacted]PKW_master.csv',
                         low_memory=False)
```

In [5]:
```
# Visualizing the first few rows of the file to ensure correct content ingesti
on.

PKW_master.head()
```

Out[5]:

	0	1	2	3	4	5	6	
0	Stockno	Value (german)	Unit (german)	Value (italian)	Unit (italian)	Value (Danish)	Unit (Danish)	
1	0102743	Isolierhülse	NaN	Guaina di isolamento	NaN	Isolationsmuffe	NaN	
2	0102759	Steckhülse	NaN	Boccola	NaN	Beskyttelsesrør	NaN	
3	0103679	Basis-Bedienpannel	NaN	Pannello operatore inverter	NaN	Basisbetjeningspanel	NaN	F cor
4	0104379	Sicherheitsschalter	NaN	Interruttore interblocco con classificazione d...	NaN	Sikkerheds endestopkontakt	NaN	inte de

5 rows × 31 columns

The first information I want to check is whether there are enough entries captured by the function `concatenate()` . If the number is significantly lower than 1 milion (roughly the number of products in the web at the time of writing), then the function did not work proprely.

In [6]:
```
PKW_master.shape
```

Out[6]:
```
(1147967, 31)
```

It looks like all data have been properly concatenated.

1.2 Cleaning the data in the file

Before loading the data into SQL, I want to be sure that the dataset is clean and free of useless duplicates.

In [7]:
```
# Checking the columns name, I can see that they need to be changed.

PKW_master.columns
```

Out[7]:
```
Index(['0', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10', '11', '12',
       '13', '14', '15', '16', '17', '18', '19', '20', '21', '22', '23', '2
4',
       '25', '26', '27', '28', '29', '30'],
      dtype='object')
```

In [8]:
```
# Therefore, I am resetting the headers to read like the first row, which is t
he header of the .csv file.

PKW_master.columns = PKW_master.iloc[0]
PKW_master.columns
```

Out[8]:
```
Index(['Stockno', 'Value (german)', 'Unit (german)', 'Value (italian)',
       'Unit (italian)', 'Value (Danish)', 'Unit (Danish)', 'Value (french)',
       'Unit (french)', 'Value (Japanese)', 'Unit (Japanese)',
       'Value (Spanish)', 'Unit (Spanish)', 'Value (Chinese)',
       'Unit (Chinese)', 'Value (English (Inter))', 'Unit (English (Inter))',
       'Value (English (Asia))', 'Unit (English (Asia))', 'Value (polish)',
       'Unit (polish)', 'Value (czech)', 'Unit (czech)', 'Value (hungarian)',
       'Unit (hungarian)', 'Value (en)', 'Unit (en)', 'Value (Russian)',
       'Unit (Russian)', 'Value (Turkish)', 'Unit (Turkish)'],
      dtype='object', name=0)
```

In [9]:
```
# Checking if the changes have worked properly.

PKW_master.head(3)
```

Out[9]:

	Stockno	Value (german)	Unit (german)	Value (italian)	Unit (italian)	Value (Danish)	Unit (Danish)	Value (french)	Unit (french)
0	Stockno	Value (german)	Unit (german)	Value (italian)	Unit (italian)	Value (Danish)	Unit (Danish)	Value (french)	Unit (french)
1	0102743	Isolierhülse	NaN	Guaina di isolamento	NaN	Isolationsmuffe	NaN	Gaine isolante	NaN
2	0102759	Steckhülse	NaN	Boccola	NaN	Beskyttelsesrør	NaN	Ferrule	NaN

3 rows × 31 columns

It is immediately clear that we have more rows that contain the header, therefore I am going to eliminate all such duplicates with the following code.

In [10]:
```
PKW_master = PKW_master[~PKW_master['Stockno'].str.contains('Stockno')]
```

In [11]:
```
# Checking if the changes have worked properly.

PKW_master.head(3)
```

Out[11]:

	Stockno	Value (german)	Unit (german)	Value (italian)	Unit (italian)	Value (Danish)	Unit (Danish)	Value (french)
1	0102743	Isolierhülse	NaN	Guaina di isolamento	NaN	Isolationsmuffe	NaN	Gaine isolante
2	0102759	Steckhülse	NaN	Boccola	NaN	Beskyttelsesrør	NaN	Ferrule
3	0103679	Basis-Bedienpannel	NaN	Pannello operatore inverter	NaN	Basisbetjeningspanel	NaN	Panneau de commande

3 rows × 31 columns

Note: I am aware that the "Unit" columns seem to be only populated by NaN values only, but in order to maintain the structure of the original dataset I am not going to get rid of those.

The most important check in this file is about duplicates in the column `Stockno` . The following code checks for this.

In [12]:
```
stockno_only = PKW_master['Stockno']
duplicates = stockno_only[stockno_only.isin(stockno_only[stockno_only.duplicat
ed()])]
duplicates.count()
```

Out[12]:
```
290
```

I need to drop the duplicates found by the code. To maintain the first instance, I use the attribute `keep=first` .

In [13]:
```
PKW_master['Stockno'].drop_duplicates(keep = 'first', inplace=True)
```

In [14]:
```
# Checking that the previous code has worked correctly.

duplicates = stockno_only[stockno_only.isin(stockno_only[stockno_only.duplicat
ed()])]
duplicates.count()
```

Out[14]:
```
0
```

Now I need to overwrite the **PKW_master.csv** file with the cleaned version that I just prepared. To do so, I will use the function `to_csv()` that migrates data from the Jupyter Notebook Environment to a `.csv` file.

In [15]:
```
PKW_master.to_csv('[redacted]
[redacted]PKW_master.csv',
                  sep='\t', encoding='utf-8', index=False)
```

2 Updating SQL

Now that we have a clean**PKW_master.csv** file, we can export it into SQL. Doing so will allow us to query the database and obtain a list of live products to base our report upon. If we missed this step, we would end up with many thousands of products that have been discontinued but will appear nonetheless in our reports as noise.

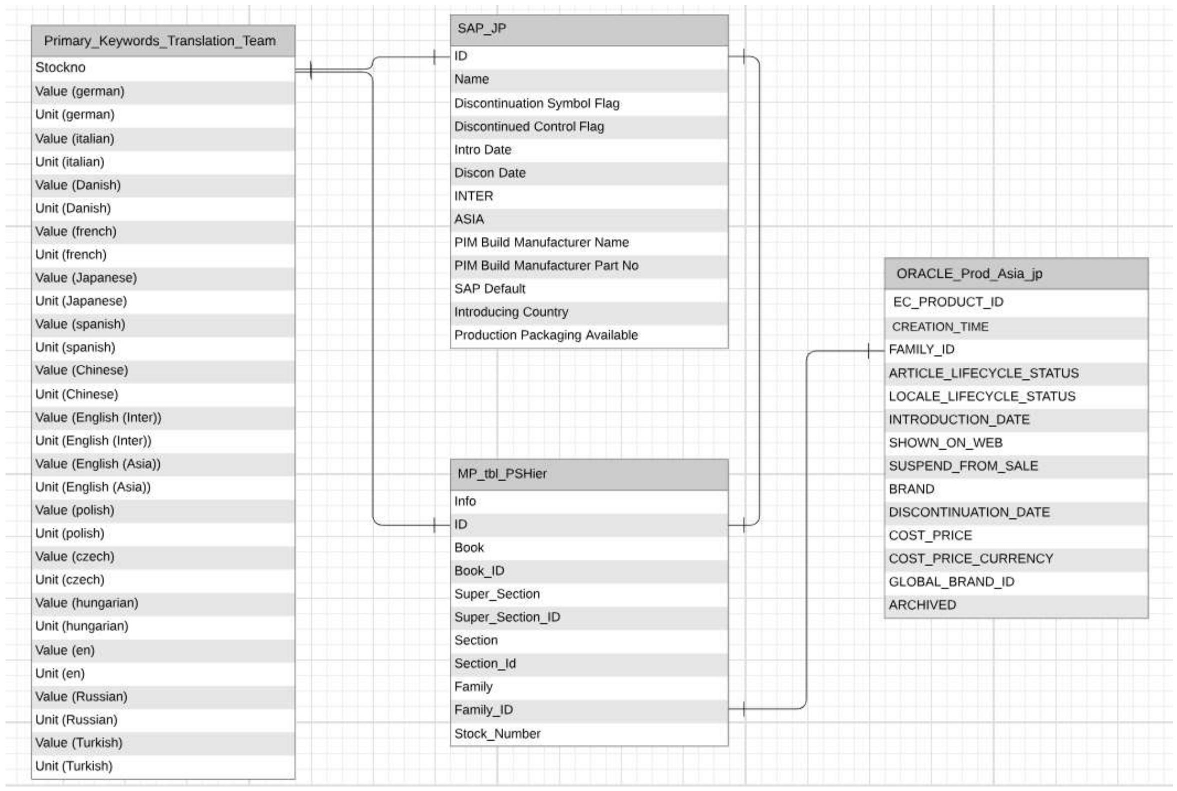
2.1 Creating a new table and linking it to the rest of the database

I want to create a new table called `Primary_Keywords_Translation_Team` , in order to query the database for information relevant to my department.

I present here the ER diagram of my newly created tab (on the left of the diagram) and how it relates to other tabs in the database that we use to query to obtain information. The SQL database in our Company contains hundreds of tables, but I chose to show here only those that directly interact with my table, and are useful to the present project.

In [16]:
```
Image(r'[redacted][redacted]\ER_d
iagram.png')
```

Out[16]:



The following is the code I use to create the table `Primary_Keywords_Translation_Team` in Microsoft SQL Server. To prevent data quality degradation over time, I set the stock number (i.e. `Stockno`) as primary key, being a unique identifier in the system.

```
CREATE TABLE [dbo].[Primary_Keywords_Translation_Team](
    [Stockno] [nvarchar](9) NOT NULL PRIMARY KEY,
    [Value (german)] [nvarchar](255) NULL,
    [Unit (german)] [nvarchar](50) NULL,
    [Value (italian)] [nvarchar](255) NULL,
    [Unit (italian)] [nvarchar](50) NULL,
    [Value (Danish)] [nvarchar](255) NULL,
    [Unit (Danish)] [nvarchar](50) NULL,
    [Value (french)] [nvarchar](255) NULL,
    [Unit (french)] [nvarchar](50) NULL,
    [Value (Japanese)] [nvarchar](255) NULL,
    [Unit (Japanese)] [nvarchar](50) NULL,
    [Value (Spanish)] [nvarchar](255) NULL,
    [Unit (Spanish)] [nvarchar](50) NULL,
    [Value (Chinese)] [nvarchar](255) NULL,
    [Unit (Chinese)] [nvarchar](50) NULL,
    [Value (English (Inter))] [nvarchar](255) NULL,
    [Unit (English (Inter))] [nvarchar](50) NULL,
    [Value (English (Asia))] [nvarchar](255) NULL,
    [Unit (English (Asia))] [nvarchar](50) NULL,
    [Value (polish)] [nvarchar](255) NULL,
    [Unit (polish)] [nvarchar](50) NULL,
    [Value (czech)] [nvarchar](255) NULL,
    [Unit (czech)] [nvarchar](50) NULL,
    [Value (hungarian)] [nvarchar](255) NULL,
    [Unit (hungarian)] [nvarchar](50) NULL,
    [Value (en)] [nvarchar](255) NULL,
    [Unit (en)] [nvarchar](50) NULL,
    [Value (Russian)] [nvarchar](255) NULL,
    [Unit (Russian)] [nvarchar](50) NULL,
    [Value (Turkish)] [nvarchar](255) NULL,
    [Unit (Turkish)] [nvarchar](50) NULL
) ON [PRIMARY]
```

2.2 Populating the table with data

Now I need to populate the newly created table with the data gathered in the PKW master file.

The following code first gets rid of any data already present in the table (if applicable), then it will insert everything found in the **PKW_master.csv** to the table.

```
truncate table Primary_Keywords_Translation_Team
BULK
INSERT Primary_Keywords_Translation_Team
FROM '\\[redacted]\PKW_master.csv'
WITH
(FIRSTROW = 1,
CODEPAGE='65001',
FIELDTERMINATOR = '\t',
ROWTERMINATOR= '\n'
)
GO
```

3. Preparing the final report

Now that all data are ready, I need to query the database to obtain my final report. However, it is best practise for us to always have a 7 digit stock number, or to add a `0` in front of a 6 digits stock number.

With the following SQL code, I am asking the system to add a `0` in front of every stock number that has less than 7 digits.

```
UPDATE Primary_Keywords_Translation_Team
SET Stockno = '0' + Stockno
WHERE LEN(Stockno)<7;
```

Finally, the query in this sequence combines all the information we need from 4 different tables:

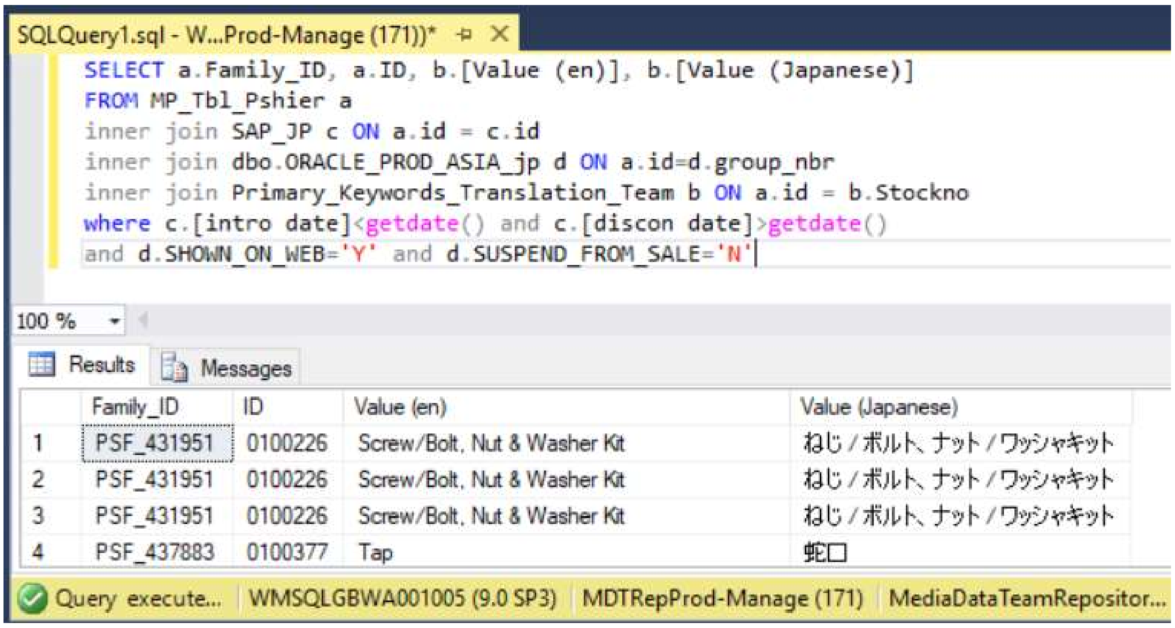
- From the `table MP_Tbl_Pshier` joined with the table `SAP_JP` , we will obtain a list of products divided by family ID.
- The `SAP_JP` table will filter for us only those products that are present in the web (i.e. live products).
- The `ORACLE_PRO_ASIA_JP` table will help filter the live products, excluding those that are suspended from sale (i.e. not available to our customers).
- The query on the PKW table will give us all PKWs for Japan that are connected with live products available for sale.

```
SELECT a.Family_ID, a.ID, b.[Value (en)], b.[Value (Japanese)]
FROM MP_Tbl_Pshier a
inner join SAP_JP c ON a.id = c.id
inner join ORACLE_PROD_ASIA_jp d ON a.id=d.group_nbr
inner join Primary_Keywords_Translation_Team b ON a.id = b.Stockno
where c.[intro date]<getdate() and c.[discon date]>getdate()
and d.SHOWN_ON_WEB='Y' and d.SUSPEND_FROM_SALE='N'
```

In [17]:
```
# As shown in the following image, the query has successfully created my repor
t.

Image('[redacted]
[redacted]sql_screenshot_final.png')
```

Out[17]:



SQLQuery1.sql - W...Prod-Manage (171))*

```
SELECT a.Family_ID, a.ID, b.[Value (en)], b.[Value (Japanese)]
FROM MP_Tbl_Pshier a
inner join SAP_JP c ON a.id = c.id
inner join dbo.ORACLE_PROD_ASIA_jp d ON a.id=d.group_nbr
inner join Primary_Keywords_Translation_Team b ON a.id = b.Stockno
where c.[intro date]<getdate() and c.[discon date]>getdate()
and d.SHOWN_ON_WEB='Y' and d.SUSPEND_FROM_SALE='N'
```

100 %

Results Messages

	Family_ID	ID	Value (en)	Value (Japanese)
1	PSF_431951	0100226	Screw/Bolt, Nut & Washer Kit	ねじ / ボルト、ナット / ワッシャキット
2	PSF_431951	0100226	Screw/Bolt, Nut & Washer Kit	ねじ / ボルト、ナット / ワッシャキット
3	PSF_431951	0100226	Screw/Bolt, Nut & Washer Kit	ねじ / ボルト、ナット / ワッシャキット
4	PSF_437883	0100377	Tap	蛇口

Query execute... | WMSQLGBWA001005 (9.0 SP3) | MDTRepProd-Manage (171) | MediaDataTeamRepositor...

The report is now ready to be exported to a spreadsheet and be presented to my Line Manager.