Insertion des produits dans la table Product

Import des bibliothèques

Entrée [ ]:
```
from lxml import etree
import pandas as pd
import mysql.connector
from mysql.connector import Error
import re
```

Parsage du catalogue REF

Entrée [ ]:
```
# à modifier avant chaque traitement d'un nouveau fichier XML
refPath = 'unzipped_files/cat-ref-FR3787ED_2019-11-22 11.30.32-22-11-2019_11-48-22'

xtree = etree.parse(refPath)
xroot = xtree.getroot()
```

Génération d'un dataframe

Entrée [ ]:
```
df_cols = ["barCode",
             "brand-id",
             "discountClass",
             "endOfLifeProduct",
             "furtherDescription",
             "height",
             "ispartialshippingallowed",
             "isremainingonbackorderallowed",
             "length",
             "longDescription_1",
             "longDescription_2",
             "longDescription_3",
             "productId",
             "publicPriceHT",
             "publicPriceTTC",
             "salesMultiple",
             "shortDescription_1",
             "shortDescription_2",
             "shortDescription_3",
             "volume",
             "weight",
             "width",
             "cat_parent"
             ]

rows = []
```

Entrée [ ]:
```
for products in xroot.iter('products'):
    for a,ec in enumerate(products.getchildren()):
        products.getchildren()[a].attrib['cat_parent'] = str(products.getparent().get('

        rows.append({ "barCode": ec.attrib['barCode'],
                      "brand-id": ec.attrib['brand-id'],
                      "discountClass": ec.attrib['discountClass'],
                      "endOfLifeProduct": ec.attrib['endOfLifeProduct'],
                      "furtherDescription": ec.attrib['furtherDescription'],
                      "height": ec.attrib['height'],
                      "ispartialshippingallowed": ec.attrib['ispartialshippingallowed
                      "isremainingonbackorderallowed": ec.attrib['isremainingonbackor
                      "length": ec.attrib['length'],
                      "longDescription_1": ec.attrib['longDescription_1'],
                      "longDescription_2": ec.attrib['longDescription_2'],
                      "longDescription_3": ec.attrib['longDescription_3'],
                      "productId": ec.attrib['productId'],
                      "publicPriceHT": ec.attrib['publicPriceHT'],
                      "publicPriceTTC": ec.attrib['publicPriceTTC'],
                      "salesMultiple": ec.attrib['salesMultiple'],
                      "shortDescription_1": ec.attrib['shortDescription_1'],
                      "shortDescription_2": ec.attrib['shortDescription_2'],
                      "shortDescription_3": ec.attrib['shortDescription_3'],
                      "volume": ec.attrib['volume'],
                      "weight": ec.attrib['weight'],
                      "width": ec.attrib['width'],
                      "cat_parent": products.getchildren()[a].attrib['cat_parent'],
                      })

dfProduct = pd.DataFrame(rows, columns=df_cols)
dfProduct.head()
```

Entrée [ ]:
```
dfProduct.shape
```

Entrée [ ]:
```
dfProduct.columns
```

NETTOYAGE DU DATAFRAME

1 - Recherche des doublons sur clé primaire productId

Entrée [ ]:
```
# dfProduct comporte 185240 lignes
resultat = dfProduct.groupby('productId').nunique()
resultat.shape
```

Entrée [ ]:
```
dfDoublon = dfProduct.groupby(['productId'])['cat_parent'] \
                             .count() \
                             .reset_index(name='count') \
                             .sort_values(['count'], ascending=False)

dfDoublon
```

Entrée [ ]:
```
dfDoublon = dfDoublon.set_index("count")
dfDoublon = dfDoublon.drop(1, axis=0)
dfDoublon
#dfProduct.set_index("productId")
```

Entrée [ ]:
```
# dfProduct = dfProduct.set_index("productId")
# for i in enumerate(dfDoublon['productId']):
#     print(i[1])
```

Entrée [ ]:
```
dfProduct = dfProduct.set_index("productId")

for i in enumerate(dfDoublon['productId']):
    dfProduct = dfProduct.drop(i[1], axis=0)

dfProduct.shape
```

Entrée [ ]:
```
dfProduct = dfProduct.reset_index()
dfProduct
```

Entrée [ ]:
```
dfProduct.longDescription_1.replace("'|/"," ", regex=True, inplace=True)
dfProduct.longDescription_2.replace("'|/"," ", regex=True, inplace=True)
dfProduct.longDescription_3.replace("'|/"," ", regex=True, inplace=True)
dfProduct.shortDescription_1.replace("'|/"," ", regex=True, inplace=True)
dfProduct.shortDescription_2.replace("'|/"," ", regex=True, inplace=True)
dfProduct.shortDescription_3.replace("'|/"," ", regex=True, inplace=True)
```

Entrée [ ]:
```
dfProduct.longDescription_1.replace("\\\\","", regex=True, inplace=True)
dfProduct.longDescription_2.replace("\\\\","", regex=True, inplace=True)
dfProduct.longDescription_3.replace("\\\\","", regex=True, inplace=True)
dfProduct.shortDescription_1.replace("\\\\","", regex=True, inplace=True)
dfProduct.shortDescription_2.replace("\\\\","", regex=True, inplace=True)
dfProduct.shortDescription_3.replace("\\\\","", regex=True, inplace=True)
```

Entrée [ ]:
```
dfProduct.publicPriceTTC.replace(",",".", regex=True, inplace=True)
dfProduct.publicPriceHT.replace(",",".", regex=True, inplace=True)
```

Entrée [ ]:
```
dfProduct.endOfLifeProduct.replace("true","1", regex=True, inplace=True)
dfProduct.endOfLifeProduct.replace("false","0", regex=True, inplace=True)
dfProduct.ispartialshippingallowed.replace("true","1", regex=True, inplace=True)
dfProduct.ispartialshippingallowed.replace("false","0", regex=True, inplace=True)
dfProduct.isremainingonbackorderallowed.replace("true","1", regex=True, inplace=True)
dfProduct.isremainingonbackorderallowed.replace("false","0", regex=True, inplace=True)
dfProduct
```

Insertion des données du dataframe dans la table Product

Entrée [ ]:
```
connection_config = {
                     'host':"localhost",
                     'port': 3308,
                     'database': 'bihr_db',
                     'user': 'BASTIER',
                     'passwd': "DA2019",
                     #'autocommit': True
                     }
```

Entrée [ ]:

```
try:
    connection = mysql.connector.connect(**connection_config)

    for i in range(len(dfProduct)):
        prod = dfProduct.iloc[i]
        #print(prod['productId'])
        productInsertQuery = """INSERT INTO product (barCode, brandId, discountClass, e
       isremainingonbackorderallowed, length, longDescription_1,longDescription_2, long
       publicPriceTTC, salesMultiple, shortDescription_1, shortDescription_2, shortDesc
       parentProduct)
       VALUES (""" + """'""" +str(prod['barCode']) + """','""" + str(prod['brand-id'])

        cursor = connection.cursor()
        result = cursor.execute(productInsertQuery)

    connection.commit()
    print("Insertion datas in product table successfully ")
    cursor.close()

except mysql.connector.Error as error:
    print("Failed to insert datas in Category table : {}".format(error))

finally:
    if (connection.is_connected()):
        cursor.close()
        connection.close()
        print("MySQL connection is closed")
```

Entrée [ ]: