Insertion des dealerPrice dans la table Dealerprice

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 PRICE

Entrée [ ]:
```
# à modifier avant chaque traitement d'un nouveau fichier XML
refPath = 'unzipped_files/cat-prices-1.1.5039520-11-2019'

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

Génération d'un dataframe dfPrice

Entrée [ ]:
```
df_cols = ["dealerPriceHT", "productId", "discountRate"]
rows = []

for i,ec in enumerate(xroot.iter('price')):
    # Attention : discounRate n'est pas implémenté dans l'api de prod
    #rows.append({"dealerPriceHT": ec.attrib['dealerPriceHT'], "productId": ec.attrib[
    rows.append({"dealerPriceHT": ec.attrib['dealerPriceHT'], "productId": ec.attrib['p

dfPrice = pd.DataFrame(rows, columns=df_cols)
dfPrice
```

Suppression des 72 productId qui étaient doubles sur le fichier REF

######(TODO : Gérer les doublons REF-Product pour supprimer ce traitement)#####

Entrée [ ]:
```
# à modifier avant chaque traitement d'un nouveau fichier XML
refPath1 = 'unzipped_files/cat-ref-FR3787ED_2019-11-22 11.30.32-22-11-2019_11-48-22'

xtree1 = etree.parse(refPath1)
xroot1 = xtree1.getroot()
```

Entrée [ ]:
```
df_cols1 = ["productId"]
rows1 = []
for products in xroot1.iter('products'):
    for a,ec in enumerate(products.getchildren()):
        rows1.append(ec.attrib['productId'])
print(len(rows1))
```

Entrée [ ]:
```
rows2 = []
doublons = []
for i in rows1:
    if i not in rows2:
        rows2.append(i)
    else:
        doublons.append(i)
print(len(rows2))
print(len(doublons))
```

Entrée [ ]:
```
# solution 1 :
rows2 = list(set(rows2) - set(doublons))
print(len(rows2))
```

Entrée [ ]:
```
# solution 2 :
# for i in doublons:
#     if i in rows2:
#         rows2.remove(i)
# print(len(rows2))
```

Entrée [ ]:
```
dfPrice['productId']
```

Entrée [ ]:
```
integrite = pd.Series(dfPrice['productId'].isin(rows2))
integrite
```

Creation de la colonne Intégrité dans le dataframe dfPrice

Entrée [ ]:
```
dfPrice['Integrité'] = integrite
dfPrice
```

Remplacement des , par . dans colonne dealerPriceHT

Entrée [ ]:
```
dfPrice.dealerPriceHT.replace(",",".", regex=True, inplace=True)
dfPrice
```

Insertion des données du dataframe dans la table Price

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(dfPrice.shape[0]):
        price = dfPrice.iloc[i]
        #print(price['Integrité'])
        #print(price['productId'])
        #print("--"+str(i)+"--")
        if price['Integrité'] == True :
            priceInsertQuery = """INSERT INTO dealerprice (dealerPriceHT ,productId, di
                           VALUES
                           ("""+ """'""" + str(price['dealerPriceHT']) + """','"""+ str
            cursor = connection.cursor()
            result = cursor.execute(priceInsertQuery)
        else:
            continue

    connection.commit()
    print("Insertion datas in dealerPrice table successful ")
    cursor.close()

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

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

Entrée [ ]: