WALMART GLOBAL TECH

by Samuel Waweru

Task 4: Data Munging

@GitHub: *https://github.com/samkamau81*

Task:

Part 1: Get the data

First, you need to get your hands on the relevant data. The shipping department has been kind enough to provide you with a repository containing all of their spreadsheets, as well as a copy of the SQLite database. First, fork and clone the repository at: https://github.com/theforage/forage-walmart-task-4

Part 2: Populate the database

Your task is to insert all of the data contained in the provided spreadsheets into the SQLite database. You will write a Python script which:

- *Reads each row from the spreadsheets.*
- *Extracts the relevant data.*
- *Munges it into a format that fits the database schema.*
- *Inserts the data into the database.*

Spreadsheet 0 is self-contained and can simply be inserted into the database, but spreadsheets 1 and 2 are dependent on one another. Spreadsheet 1 contains a single product per row, you will need to combine each row based on its shipping identifier, determine the quantity of goods in the shipment, and add a new row to the database for each product in the shipment. The origin and destination for each shipment in spreadsheet 1 are contained in spreadsheet 2. You may assume that all the given data is valid - product names are always spelled the same way, quantities are positive, etc. When you're finished, convert the python script you used to populate the database into a PDF and submit it below.

['origin_warehouse', 'destination_store', 'product', 'on_time', 'product_quantity', 'driver_identifier']

['shipment_identifier', 'origin_warehouse', 'destination_store', 'driver_identifier']

['shipment_identifier', 'product', 'on_time']

['shipment_identifier', 'product', 'on_time', 'origin_warehouse', 'destination_store', 'driver_identifier']

To accomplish the task;

First, I need to open and read the spreadsheets using a Python library such as pandas. I used the pandas.read_excel() function to read the Excel files into a pandas Data Frame.

Next, I need to extract the relevant data from each spreadsheet. I did this by selecting specific columns or rows using the DataFrame's indexing and slicing capabilities.

Once I have extracted the data, I munged it into a format that fits the database schema. This may involve transforming the data in some way, such as converting columns to a specific data type or applying a function to each value.

Finally, I used the Python sqlite3 module to connect to the SQLite database and insert the data.

Here is the code in Python;

```
import pandas as pd
import csv, sys
import sqlite3


data0 =pd.read_csv( 'shipping_data_0.csv')
data1=pd.read_csv("shipping_data_1.csv")
data2=pd.read_csv("shipping_data_2.csv")

print(data0.head(5),'\n',data1.head(5),'\n',data2.head(5),'\n')
print(data1.columns)

origin_warehouse = []
destination_store = []
driver_identifier = []

for i in data1['shipment_identifier']:
    count = 0
    for j in data2['shipment_identifier']:
        if i==j:
            origin_warehouse.append(data2.iloc[count, 1])
            destination_store.append(data2.iloc[count, 2])
            driver_identifier.append(data2.iloc[count, 3])
        count+=1


print(origin_warehouse,'\n',destination_store,'\n',driver_identifier)

data1["origin_warehouse"]=origin_warehouse
data1["destination_store"]=destination_store
data1["driver_identifier"]=driver_identifier


print(data1.columns)
print(data1["origin_warehouse"],data1["destination_store"],data1["driver_identifier"])

data1.to_csv("data1.csv", index=False)


'''creating the shipment details table'''
conn = sqlite3.connect ("shipment_database.db")
c = conn.cursor ()
#Creating Table 1 from shipping_data0
c.execute ("CREATE TABLE IF NOT EXISTS shipment_Details(origin_warehouse
TEXT,destination_store TEXT, product TEXT, on_time BOOLEAN, product_quantity INTEGER,
driver_identifier TEXT NOT NULL PRIMARY KEY )")
#creating Table 2 from combining shipping data1 and shipping data2 to form data1.csv
c.execute ("CREATE TABLE IF NOT EXISTS shipment_Details1(shipment_identifier
TEXT,product TEXT, on_time BOOLEAN, origin_warehouse TEXT, destination_store TEXT,
driver_identifier TEXT, FOREIGN KEY(driver_identifier) REFERENCES
shipment_Details(driver_identifier))")
conn.commit ()
conn.close ()
```

```
with open('shipping_data_0.csv', newline='') as f:
    reader = csv.reader(f)
    try:
        for row in reader:
            print(row)
            conn = sqlite3.connect("shipment_database.db")
            cur = conn.cursor()
            cur.execute("INSERT INTO shipment_Details VALUES(?, ?, ? ,? ,? ,?)", row)
            conn.commit()
    except csv.Error as e:
        sys.exit('file {}, line {}: {}'.format('shipping_data_0.csv', reader.line_num,
e))

with open('data1.csv', newline='') as f:
    reader = csv.reader(f)
    try:
        for row in reader:
            print(row)
            conn = sqlite3.connect("shipment_database.db")
            cur = conn.cursor()
            cur.execute("INSERT INTO shipment_Details1 VALUES(?, ?, ? ,? ,? ,?)", row)
            conn.commit()
    except csv.Error as e:
        sys.exit('file {}, line {}: {}'.format('data1.csv', reader.line_num, e))

conn = sqlite3.connect ("shipment_database.db")
c = conn.cursor ()
c.execute(" SELECT shipment_Details1.shipment_identifier,
shipment_Details.product_quantity FROM shipment_Details1 INNER JOIN shipment_Details
ON shipment_Details1.driver_identifier=shipment_Details.driver_identifier;")
conn.commit ()
```