Task 4: Data Munging

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 their spreadsheets, as well as a copy of the SQLite database. First, fork and clone the repository at: https://github.com/Sagargwu/Walmart-By-Forage-Internship/tree/main/Task%204%20Data%20Munging

Part 2: Populate the database
Your task is to insert all 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.

Solution

```
import csv
import sqlite3

def create_tables(cursor):
    cursor.execute("""
        CREATE TABLE IF NOT EXISTS shipping_data_0 (
            origin_warehouse TEXT,
            destination_store TEXT,
            product TEXT,
            on_time TEXT,
            product_quantity INTEGER,
```

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 their spreadsheets, as well as a copy of the SQLite database. First, fork and clone the repository at:
https://github.com/Sagargwu/Walmart-By-Forage-Internship.git

Part 2: Populate the database

- 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.

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

```
        driver_identifier TEXT
    )
    """)

    cursor.execute("""
    CREATE TABLE IF NOT EXISTS shipping_data_1 (        shipment_identifier TEXT,
product TEXT,        on_time TEXT,        origin_warehouse TEXT,
destination_store TEXT
    )    """)  def insert_shipping_data_0(cursor):
with open('data/shipping_data_0.csv', 'r') as file:
    csv_reader = csv.reader(file)
next(csv_reader)        for row in csv_reader:
        origin_warehouse, destination_store, product, on_time, product_quantity,
driver_identifier = row        cursor.execute("INSERT INTO shipping_data_0
(origin_warehouse, destination_store, product, on_time, product_quantity, driver_identifier)
VALUES (?, ?, ?, ?, ?, ?)",
                    (origin_warehouse, destination_store, product, on_time,
product_quantity, driver_identifier))  def insert_shipping_data_2(cursor):    with
open('data/shipping_data_2.csv', 'r') as file:        csv_reader = csv.reader(file)
next(csv_reader)        shipping_data_2_rows = [row for row in
csv_reader]
    with open('data/shipping_data_1.csv', 'r') as file:
    csv_reader = csv.reader(file)        next(csv_reader)
for row in csv_reader:
        shipment_identifier, product, on_time = row        matching_rows = [r for r in
shipping_data_2_rows if r[0] == shipment_identifier]        if matching_rows:
            origin_warehouse, destination_store, driver_identifier = matching_rows[0][1],
matching_rows[0][2], matching_rows[0][3]            cursor.execute("INSERT INTO
shipping_data_1 (shipment_identifier, product, on_time, origin_warehouse, destination_store)
VALUES (?, ?, ?, ?, ?)",
```

```
                    (shipment_identifier, product, on_time, origin_warehouse,
destination_store))  if __name__ == "__main__":    conn =
sqlite3.connect('shipment_database.db')    cursor
= conn.cursor()      create_tables(cursor)  # Create the
necessary tables
insert_shipping_data_0(cursor)    insert_shipping_data_2(cursor)

conn.commit()    conn.close()
```