```
import csv
import sqlite3

class DatabaseConnector:
    def __init__(self, database_file):
        self.connection = sqlite3.connect(database_file)
        self.cursor = self.connection.cursor()

    def populate(self, spreadsheet_folder):

        with open(f"{spreadsheet_folder}/shipping_data_0.csv", "r") as spreadsheet_file_0:
            with open(f"{spreadsheet_folder}/shipping_data_1.csv", "r") as spreadsheet_file_1:
                with open(f"{spreadsheet_folder}/shipping_data_2.csv", "r") as spreadsheet_file_2:

                    csv_reader_0 = csv.reader(spreadsheet_file_0)
                    csv_reader_1 = csv.reader(spreadsheet_file_1)
                    csv_reader_2 = csv.reader(spreadsheet_file_2)

                    self.populate_first_shipping_data(csv_reader_0)
                    self.populate_second_shipping_data(csv_reader_1, csv_reader_2)

    def populate_first_shipping_data(self, csv_reader_0):

        for row_index, row in enumerate(csv_reader_0):

            if row_index > 0:

                product_name = row[2]
                product_quantity = row[4]
                origin = row[0]
                destination = row[1]
                # Insert the data into the database
                self.insert_product_if_it_does_not_already_exist(product_name)
                self.insert_shipment(product_name, product_quantity, origin, destination)
                print(f"Inserted product {row_index} from shipping_data_0")

    def populate_second_shipping_data(self, csv_reader_1, csv_reader_2):

        shipment_info = {}


        for row_index, row in enumerate(csv_reader_2):

            if row_index > 0:

                shipment_identifier = row[0]
                origin = row[1]
                destination = row[2]

                shipment_info[shipment_identifier] = {
                    "origin": origin,
                    "destination": destination,
                    "products": {}
                }

        # Read product data from shipping_data_1
        for row_index, row in enumerate(csv_reader_1):

            if row_index > 0:

                shipment_identifier = row[0]
                product_name = row[1]

                products = shipment_info[shipment_identifier]["products"]
                if products.get(product_name, None) is None:
                    products[product_name] = 1
```

```python
                else:
                    products[product_name] += 1

        # Insert the data into the database
        count = 0
        for shipment_identifier, shipment in shipment_info.items():

            origin = shipment["origin"]
            destination = shipment["destination"]
            for product_name, product_quantity in shipment["products"].items():
                # Insert products into the database
                self.insert_product_if_it_does_not_already_exist(product_name)
                self.insert_shipment(product_name, product_quantity, origin, destination)

                print(f"Inserted product {count} from shipping_data_1")
                count += 1

    def insert_product_if_it_does_not_already_exist(self, product_name):
        '''Insert a new product into the database if it does not already exist.'''
        query = """
        INSERT OR IGNORE INTO product (name)
        VALUES (?);
        """
        self.cursor.execute(query, (product_name,))
        self.connection.commit()

    def insert_shipment(self, product_name, product_quantity, origin, destination):
        '''Insert a new shipment into the database.'''
        # Collect the product id
        query = """
        SELECT id FROM product WHERE name = ?;
        """
        self.cursor.execute(query, (product_name,))
        product_id = self.cursor.fetchone()[0]
        # Insert the shipment
        query = """
        INSERT OR IGNORE INTO shipment (product_id, quantity, origin, destination)
        VALUES (?, ?, ?, ?);
        """
        self.cursor.execute(query, (product_id, product_quantity, origin, destination))
        self.connection.commit()

    def close(self):
        '''Close the database connection.'''
        self.connection.close()

if __name__ == '__main__':
    # Create a DatabaseConnector object
    database_connector = DatabaseConnector("shipment_database.db")
    # Populate the database with the data from the spreadsheet
    database_connector.populate("./data")
    # Close the database connection
    database_connector.close()
```