```
import sqlite3
import csv
import os

class DatabaseConnector:
    """ Manages a connection to a SQLite database and populates it with data from CSV files."""

    def __init__(self, database_file):
        # Initialize the database connection and create tables
        self.connection = sqlite3.connect(database_file)
        self.cursor = self.connection.cursor()
        self.create_tables()

    def create_tables(self):
        """ Creates necessary tables in the database if they don’t already exist. """
        # Create product table
        self.cursor.execute('''
            CREATE TABLE IF NOT EXISTS product (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                name TEXT UNIQUE
            );
        ''')
        # Create shipment table
        self.cursor.execute('''
            CREATE TABLE IF NOT EXISTS shipment (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                product_id INTEGER,
                quantity INTEGER,
                origin TEXT,
                destination TEXT,
                FOREIGN KEY (product_id) REFERENCES product(id)
            );
        ''')
        self.connection.commit()

    def populate(self, folder):
        """ Populates the database with data from the CSV files in the specified folder. """
        file_paths = [
            os.path.join(folder, "shipping_data_0.csv"),
            os.path.join(folder, "shipping_data_1.csv"),
            os.path.join(folder, "shipping_data_2.csv")
        ]

        try:
            # Open all CSV files
            with open(file_paths[0], newline='') as file_0, \
                 open(file_paths[1], newline='') as file_1, \
                 open(file_paths[2], newline='') as file_2:

                reader_0 = csv.reader(file_0)
                reader_1 = csv.reader(file_1)
                reader_2 = csv.reader(file_2)

                # Populate data from the CSV files
                self.populate_shipping_data_1(reader_0)
```

```
            self.populate_shipping_data_2(reader_1, reader_2)

        except FileNotFoundError as e:
            print(f"Error: {e}. Ensure the CSV files exist in the specified folder.")
        except Exception as e:
            print(f"An error occurred: {e}")

    def populate_shipping_data_1(self, reader_0):
        """ Populates the database with data from the first CSV file. """
        for row_idx, row in enumerate(reader_0):
            if row_idx > 0:
                product_name = row[2]
                try:
                    product_quantity = int(row[4])  # Convert quantity to integer
                except ValueError:
                    print(f"Warning: Invalid quantity '{row[4]}' in row {row_idx}. Skipping.")
                    continue
                origin = row[0]
                destination = row[1]

                self.insert_product(product_name)
                self.insert_shipment(product_name, product_quantity, origin, destination)

    def populate_shipping_data_2(self, reader_1, reader_2):
        """ Populates the database with data from the second and third CSV files. """
        shipment_info = {}

        # Read and store shipment information
        for row_idx, row in enumerate(reader_2):
            if row_idx > 0:
                shipment_identifier = row[0]
                shipment_info[shipment_identifier] = {
                    "origin": row[1],
                    "destination": row[2],
                    "products": {}
                }

        # Aggregate product data for each shipment
        for row_idx, row in enumerate(reader_1):
            if row_idx > 0:
                shipment_identifier = row[0]
                product_name = row[1]
                if shipment_identifier in shipment_info:
                    products = shipment_info[shipment_identifier]["products"]
                    products[product_name] = products.get(product_name, 0) + 1

        # Insert aggregated product and shipment data
        for shipment in shipment_info.values():
            for product_name, product_quantity in shipment["products"].items():
                self.insert_product(product_name)
                self.insert_shipment(product_name, product_quantity, shipment["origin"],
shipment["destination"])

    def insert_product(self, product_name):
        """ Inserts a new product into the database if it does not already exist. """
```

```
        query = 'INSERT OR IGNORE INTO product(name) VALUES(?);'
        try:
            self.cursor.execute(query, (product_name,))
            self.connection.commit()
        except sqlite3.Error as e:
            print(f"Error inserting product '{product_name}': {e}")

    def insert_shipment(self, product_name, product_quantity, origin, destination):
        """ Inserts a new shipment into the database. """
        query = 'SELECT id FROM product WHERE name = ?;'
        self.cursor.execute(query, (product_name,))
        result = self.cursor.fetchone()

        if result:
            product_id = result[0]
            query = 'INSERT INTO shipment(product_id, quantity, origin, destination) VALUES(?,?,?,?);'
            try:
                self.cursor.execute(query, (product_id, product_quantity, origin, destination))
                self.connection.commit()
            except sqlite3.Error as e:
                print(f"Error inserting shipment for product '{product_name}': {e}")
        else:
            print(f"Warning: Product '{product_name}' not found in the database.")

    def close(self):
        """ Closes the database connection. """
        self.connection.close()

if __name__ == '__main__':
    db_connector = DatabaseConnector("shipment_database.db")
    db_connector.populate("./data")
    db_connector.close()
```