```
import csv
import sqlite3
import pandas as pd


def getDataFrame(fileData):
  with open(fileData, newline = '') as csvData:
    myData = csv.reader(csvData, delimiter=',')
    dataFrame = pd.DataFrame(myData)
    return dataFrame


connection = sqlite3.connect('/shipment_database.db')
cur = connection.cursor()


#columns origin_warehouse, destination_store, product, on_time, product_quantity, driver_identifier
dataFrame = getDataFrame('/shipping_data_0.csv')
csvData1 = getDataFrame('/shipping_data_1.csv')
csvData2 = getDataFrame('/shipping_data_2.csv')


products = dataFrame[2][1:]
cur.executemany("INSERT OR IGNORE INTO product (name) VALUES (?)", [(product,) for product in products])
connection.commit()


# Insert data into shipment table, referencing product IDs
for index, row in dataFrame.iterrows():
  if index > 0:
    product_name = row[2]
    cur.execute("SELECT id FROM product WHERE name = ?", (product_name,))
    product_id = cur.fetchone()
    if product_id:
      product_id = product_id[0]
      try:
        cur.execute("""
              INSERT INTO shipment (product_id, quantity, origin, destination)
              VALUES (?, ?, ?, ?)
          """, (product_id, row[4], row[0], row[1]))
      except Exception as e:
          print(f"Error inserting row {index}: {e}")
    else:
        print(f"Product '{product_name}' not found in product table")
connection.commit()


shipment_id = {}
shippings = {}

for index,row in csvData1.iterrows():
  if index > 0:
    product_name = row[1]

    if row[0] in shipment_id:

      if product_name in shipment_id[row[0]]:
        shipment_id[row[0]][product_name] += 1
      else:
        shipment_id[row[0]][product_name] = 1

    else:
      shipment_id[row[0]] = {product_name: 1}

    cur.execute("INSERT OR IGNORE INTO product (name) VALUES (?)", (product_name,))

connection.commit()

for index,row in csvData2.iterrows():
  if index > 0:
    ship_id = row[0]
    if ship_id in shippings:
      print("shipping id already exists")
    else:
      shippings[ship_id] = [row[1],row[2]]



#TODO package dictionaries and send to database
for shippingId,products in shipment_id.items():
  for product,quantity in products.items():
    cur.execute("SELECT id FROM product WHERE name = ?", (product,))
    product_id = cur.fetchone()
```

```
      if product_id:
        product_id = product_id[0]
        try:
          cur.execute("""
                INSERT INTO shipment (product_id, quantity, origin, destination)
                VALUES (?, ?, ?, ?)
            """, (product_id, quantity, shippings[shippingId][0],shippings[shippingId][1],))
        except Exception as e:
            print(f"Error inserting shipping Id {shippingId} for {product} quantity : {quantity}: {e}")
      else:
          print(f"Product '{product_name}' not found in product table")


  connection.commit()



  cur.execute("SELECT * FROM shipment")
  rows = cur.fetchall()
  for row in rows:
    print(row)
```

```
(97, 42, 25, 'e92bc306-314e-4853-8728-0107aa27b936', 'f94fbe6a-3f7a-4141-8405-9748f55ac880')
(98, 13, 26, '333bc679-0295-49cd-ad2e-cbb89a65bd99', 'b6acb8d3-1efd-4496-b0e7-5f2d4be9ff45')
(99, 24, 40, 'b7f83fe7-6ae2-4e75-81e1-8440d26778c2', 'd72c004c-6999-4ffb-b291-1bd581c65d95')
(100, 8, 42, 'd26316d2-19e2-45c0-9ada-b92ba56a4da2', '3343a3b9-be11-4276-bca9-d7f4dd0b40aa')
(101, 5, 84, 'b65da068-cfdf-4247-84db-91232e610217', 'd74dc441-1948-4ba8-aeef-843bd074e246')
(102, 34, 14, 'fc629195-daf2-465a-91bd-330719735bb1', '60df7002-bc50-4357-8014-02e42f5a5ab6')
(103, 35, 25, 'ee0f4141-13ad-4289-a414-f1756a5748ef', '3264f6eb-ce55-44da-845b-32cbf99da950')
(104, 43, 39, 'c6d4e9d1-45a7-4948-9ec6-be687c4b8bba', 'fdc68f90-b2e4-483b-985d-b237467e8d9a')
(105, 6, 16, 'bad3d1a2-b14d-4efa-8f81-5270ee921ae6', 'd7a8eb36-e00d-484f-b21e-a2ab742f8a2a')
(106, 13, 95, 'd2ee1b75-2218-4753-9487-dcca23d667c6', '0a994581-341f-43bf-979d-ece1e58de7ec')
(107, 9, 54, '6a6d3fce-c5aa-4154-a6a3-b56cb41f709f', '403bf915-a897-4918-933b-3996e144e960')
(108, 10, 20, 'b19cec0d-357e-4c6b-9257-8be52b1c71b5', 'd3b17672-60fb-443f-a047-2c379132dcb1')
(109, 21, 7, 'd2a2460e-00d1-41f2-84cc-eba01eb88d75', 'b9f78d5b-79ae-441e-9dbf-592767af34a5')
(110, 19, 35, '75891066-59b4-437b-951f-ec553fb26b94', '28fff0d2-38ea-40a7-b2ef-c2a2f7e69370')
(111, 18, 3, 'bb75bf7d-c008-4267-bf92-6089cff5fe56', '5e9405de-a078-4b00-99c6-96564568b63c')
(112, 34, 2, 'bb75bf7d-c008-4267-bf92-6089cff5fe56', '5e9405de-a078-4b00-99c6-96564568b63c')
(113, 41, 1, '372fd2b1-b2a7-4553-b6d7-426a1bc88e56', 'e34973c8-9ca9-4a06-b497-7a8b49625fc2')
(114, 13, 3, '469d957f-28ef-4eac-956a-d2a42b06d3ab', 'fcadc756-61e9-41bb-871b-d3546c5aa981')
(115, 20, 2, '469d957f-28ef-4eac-956a-d2a42b06d3ab', 'fcadc756-61e9-41bb-871b-d3546c5aa981')
(116, 23, 1, 'cd140190-a53b-4660-a5b4-cc844a6506f0', '89ba200c-ca90-443a-b64f-397bce091eae')
(117, 1, 4, 'cd140190-a53b-4660-a5b4-cc844a6506f0', '89ba200c-ca90-443a-b64f-397bce091eae')
(118, 34, 4, 'cd140190-a53b-4660-a5b4-cc844a6506f0', '89ba200c-ca90-443a-b64f-397bce091eae')
(119, 1, 3, 'c6addf8b-eea6-43b8-9040-b5620b1a0d99', '7aebe820-8478-4a29-a606-7c59af677e24')
(120, 44, 3, 'c6addf8b-eea6-43b8-9040-b5620b1a0d99', '7aebe820-8478-4a29-a606-7c59af677e24')
(121, 15, 3, '5d64f731-cb01-4992-a27c-a6e1342f4913', 'd57d76d8-7dca-4ee4-84c0-1745fb4c8779')
(122, 14, 1, '5d64f731-cb01-4992-a27c-a6e1342f4913', 'd57d76d8-7dca-4ee4-84c0-1745fb4c8779')
(123, 17, 2, '5d64f731-cb01-4992-a27c-a6e1342f4913', 'd57d76d8-7dca-4ee4-84c0-1745fb4c8779')
(124, 45, 3, 'abc09fec-2fa0-48f6-b7c4-913620785520', '52479603-9957-4e4b-91eb-337c358d1755')
(125, 19, 1, 'abc09fec-2fa0-48f6-b7c4-913620785520', '52479603-9957-4e4b-91eb-337c358d1755')
(126, 5, 4, 'abc09fec-2fa0-48f6-b7c4-913620785520', '52479603-9957-4e4b-91eb-337c358d1755')
(127, 28, 2, 'bb53f18b-e3c5-48c1-900e-e8ed623ca467', '1add84b7-14f5-4857-903b-578408246946')
(128, 12, 2, 'bb53f18b-e3c5-48c1-900e-e8ed623ca467', '1add84b7-14f5-4857-903b-578408246946')
(129, 43, 4, 'bb53f18b-e3c5-48c1-900e-e8ed623ca467', '1add84b7-14f5-4857-903b-578408246946')
(130, 6, 1, 'ee67c3b0-aa89-4b3b-8bbc-9d70695c132b', 'fa0ce0bb-b0d8-469d-8d42-e1153fc48272')
(131, 18, 2, 'ee67c3b0-aa89-4b3b-8bbc-9d70695c132b', 'fa0ce0bb-b0d8-469d-8d42-e1153fc48272')
(132, 20, 4, 'ee67c3b0-aa89-4b3b-8bbc-9d70695c132b', 'fa0ce0bb-b0d8-469d-8d42-e1153fc48272')
(133, 35, 2, 'f04f3daf-8ede-4787-a3ad-6ff06274229d', '130208de-fef4-46cd-8b9b-1ea5b939895b')
(134, 41, 2, '48c4ca28-4db8-420e-af57-241818a81194', 'dc042557-cbee-4743-9b72-2c0a34a99cc2')
(135, 24, 1, '48c4ca28-4db8-420e-af57-241818a81194', 'dc042557-cbee-4743-9b72-2c0a34a99cc2')
(136, 15, 2, '48c4ca28-4db8-420e-af57-241818a81194', 'dc042557-cbee-4743-9b72-2c0a34a99cc2')
(137, 7, 1, '57001f3e-d6be-4031-9295-a6b208c0ad46', '49714439-c858-4e61-b7c6-f4d4c848c46b')
(138, 43, 4, 'c42221be-4851-42df-9184-1b4f96962bb7', '134fb73e-fc99-41e4-92c4-6ecbada5574f')
(139, 22, 3, 'c42221be-4851-42df-9184-1b4f96962bb7', '134fb73e-fc99-41e4-92c4-6ecbada5574f')
(140, 7, 1, 'c42221be-4851-42df-9184-1b4f96962bb7', '134fb73e-fc99-41e4-92c4-6ecbada5574f')
(141, 25, 3, '20efa3c2-c498-4908-8af4-f81c76781912', '0167b0c3-60fd-4fbd-b378-fba0ceac39ff')
(142, 28, 3, '20efa3c2-c498-4908-8af4-f81c76781912', '0167b0c3-60fd-4fbd-b378-fba0ceac39ff')
(143, 9, 2, 'f156fb67-e7e7-448c-b9c3-0b5f0425d134', '2ba952cd-d5bf-4bb4-96fe-02fd4e166d12')
(144, 19, 3, 'f156fb67-e7e7-448c-b9c3-0b5f0425d134', '2ba952cd-d5bf-4bb4-96fe-02fd4e166d12')
(145, 22, 2, 'c44107fb-70a4-4ff2-bedb-48b9ecfd3d6d', '479bfe4c-7137-44dd-bb42-a4ab7900aa24')
(146, 18, 1, '026ffd7c-cff2-4daa-ae76-6768d3283861', 'd33a7b5d-15e5-4ba6-93ef-136bbcbf4946')
(147, 33, 4, '026ffd7c-cff2-4daa-ae76-6768d3283861', 'd33a7b5d-15e5-4ba6-93ef-136bbcbf4946')
(148, 18, 5, '950db98c-fb0f-4b78-bbee-bf9ae97faeca', '5b9e2a68-7967-46e0-8e35-bc993a1f07e6')
(149, 41, 3, '950db98c-fb0f-4b78-bbee-bf9ae97faeca', '5b9e2a68-7967-46e0-8e35-bc993a1f07e6')
(150, 24, 4, 'b377e5d5-563f-475d-8c6d-b9f85ad861fd', 'fa60bc82-665e-4fe0-8f1f-b8a7675c2e2a')
(151, 10, 1, 'b377e5d5-563f-475d-8c6d-b9f85ad861fd', 'fa60bc82-665e-4fe0-8f1f-b8a7675c2e2a')
(152, 34, 4, 'b377e5d5-563f-475d-8c6d-b9f85ad861fd', 'fa60bc82-665e-4fe0-8f1f-b8a7675c2e2a')
(153, 39, 1, '5158fc84-71e0-47a1-84e9-b3e446a391ae', '76f02f30-28cd-4f15-88be-9c64860d1fce')
(154, 40, 3, '5158fc84-71e0-47a1-84e9-b3e446a391ae', '76f02f30-28cd-4f15-88be-9c64860d1fce')
```