Database Setup:

Approach 1

Single Table called shipping_data created The script connects to the SQLite database (or creates it if it doesn't exist). A table named shipping_data is created if it does not already exist. Loading Data:

The script loads the data from shipping_data_0.csv directly into the database. It then loads shipping_data_1.csv and shipping_data_2.csv for further processing. Merging and Processing:

The script merges df_1 and df_2 on the shipping_identifier column to combine product details with their corresponding origin and destination information. Data Insertion:

The processed data is then inserted into the database.

```
import pandas as pd
import sqlite3
import logging

# Configure logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

# File paths to CSV files
csv_file_0 = '/Users/shamalachandrappa/Desktop/Git Hub Push Everyday/Walmart Stock Price from 1972-2022/forage-walmart-task-4/dat
csv_file_1 = '/Users/shamalachandrappa/Desktop/Git Hub Push Everyday/Walmart Stock Price from 1972-2022/forage-walmart-task-4/dat
csv_file_2 = '/Users/shamalachandrappa/Desktop/Git Hub Push Everyday/Walmart Stock Price from 1972-2022/forage-walmart-task-4/dat

# Function to create the SQLite connection
def create_connection(db_file):
    try:
        conn = sqlite3.connect(db_file)
        logging.info("Connected to SQLite database.")
        return conn
    except sqlite3.Error as e:
        logging.error(f"Error connecting to SQLite database: {e}")
        return None

# Function to drop and create the shipping_data table
def setup_database(conn):
    try:
        cursor = conn.cursor()
        cursor.execute('DROP TABLE IF EXISTS shipping_data')
        cursor.execute('''
        CREATE TABLE shipping_data (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            shipment_id TEXT,
            product TEXT,
            quantity INTEGER,
            origin TEXT,
            destination TEXT,
            on_time TEXT,
            driver_identifier TEXT
        )
        ''')
        logging.info("Table shipping_data created successfully.")
    except sqlite3.Error as e:
        logging.error(f"Error setting up the database: {e}")

# Function to load CSV data into a DataFrame
def load_csv_data(file_path):
    try:
        df = pd.read_csv(file_path)
        logging.info(f"Loaded data from {file_path}.")
        return df
    except Exception as e:
        logging.error(f"Error loading CSV file {file_path}: {e}")
        return None

# Function to process and insert data into the database
def insert_data(conn, df_0, df_1, df_2):
    try:
        # Rename columns to fit the schema for df_0
        df_0 = df_0.rename(columns={
            'origin_warehouse': 'origin'
            'origin_warehouse': 'origin',
            'destination_store': 'destination',
            'product_quantity': 'quantity'
        })

        # Insert data from the first CSV file directly into the database
        df_0.to_sql('shipping_data', conn, if_exists='append', index=False)
        logging.info("Inserted data from shipping_data_0.csv into the database.")

        # Group the data in df_1 by shipment_id and product to calculate total quantity per product per shipment
        df_1_grouped = df_1.groupby(['shipment_identifier', 'product', 'on_time']).size().reset_index(name='quantity')

        # Merge the grouped data with df_2 to get origin and destination information
        merged_df = pd.merge(df_1_grouped, df_2, on='shipment_identifier', how='inner')

        # Create a new DataFrame to hold the combined data
        combined_data = []

        # Process each row in the merged DataFrame
        for _, row in merged_df.iterrows():
            combined_data.append({
                'shipment_id': row['shipment_identifier'],
                'product': row['product'],
                'quantity': row['quantity'],
                'origin': row['origin_warehouse'],
                'destination': row['destination_store'],
                'on_time': row['on_time'],
                'driver_identifier': row['driver_identifier']
            })

        # Convert the combined data into a DataFrame
        combined_df = pd.DataFrame(combined_data)

        # Insert the processed data into the database
        combined_df.to_sql('shipping_data', conn, if_exists='append', index=False)
        logging.info("Inserted processed data from shipping_data_1.csv and shipping_data_2.csv into the database.")
    except sqlite3.Error as e:
        logging.error(f"Error inserting data into the database: {e}")
    except Exception as e:
        logging.error(f"Unexpected error: {e}")

# Function to fetch and display data from the database
def fetch_and_display_data(conn):
    try:
        cursor = conn.cursor()
        cursor.execute('SELECT * FROM shipping_data')
        rows = cursor.fetchall()
        logging.info("Fetched data from the database.")

        # Convert fetched data into a DataFrame
        df = pd.DataFrame(rows, columns=['id', 'shipment_id', 'product', 'quantity', 'origin', 'destination', 'on_time', 'driver_

        # Display the fetched data
        print(df)

        return df
    except sqlite3.Error as e:
        logging.error(f"Error fetching data from the database: {e}")
        return None

# Function to save the fetched data to a CSV file
def save_data_to_csv(df, output_file):
    try:
        df.to_csv(output_file, index=False)
        logging.info(f"Data saved to {output_file}.")
    except Exception as e:
        logging.error(f"Error saving data to CSV file {output_file}: {e}")

# Main function to execute the steps
def main():
    # Create a database connection
    conn = create_connection('shipping_data.db')
    if conn is not None:
        # Set up the database by dropping and creating the table
        setup_database(conn)

        # Load data from CSV files
        df_0 = load_csv_data(csv_file_0)
        df_1 = load_csv_data(csv_file_1)
        df_2 = load_csv_data(csv_file_2)

        # Ensure all data is loaded successfully before proceeding
        if df_0 is not None and df_1 is not None and df_2 is not None:
            # Insert data into the database
            insert_data(conn, df_0, df_1, df_2)

            # Fetch and display the data
            fetched_df = fetch_and_display_data(conn)

            # Save the fetched data to a CSV file
            if fetched_df is not None:
                save_data_to_csv(fetched_df, 'output_shipping_data.csv')

        # Close the database connection
        conn.close()
        logging.info("Closed SQLite database connection.")
    else:
        logging.error("Failed to create database connection.")

if __name__ == "__main__":
    main()
```

```
2024-08-24 17:11:13,546 - INFO - Connected to SQLite database.
2024-08-24 17:11:13,549 - INFO - Table shipping_data created successfully.
2024-08-24 17:11:13,555 - INFO - Loaded data from /Users/shamalachandrappa/Desktop/Git Hub Push Everyday/Walmart Stock Price
2024-08-24 17:11:13,556 - INFO - Loaded data from /Users/shamalachandrappa/Desktop/Git Hub Push Everyday/Walmart Stock Price
2024-08-24 17:11:13,558 - INFO - Loaded data from /Users/shamalachandrappa/Desktop/Git Hub Push Everyday/Walmart Stock Price
2024-08-24 17:11:13,562 - INFO - Inserted data from shipping_data_0.csv into the database.
2024-08-24 17:11:13,570 - INFO - Inserted processed data from shipping_data_1.csv and shipping_data_2.csv into the database.
2024-08-24 17:11:13,571 - INFO - Fetched data from the database.
2024-08-24 17:11:13,580 - INFO - Data saved to output_shipping_data.csv.
2024-08-24 17:11:13,581 - INFO - Closed SQLite database connection.
      id                           shipment_id        product  quantity  \
0      1                                  None         lotion        59
1      2                                  None        windows        28
2      3                                  None           skis        63
3      4                                  None          bikes        47
4      5                                  None          candy        73
..   ...                                   ...            ...       ...
149  150  e31e22c1-5395-43d8-8a0a-79396d627f66          pants         2
150  151  e31e22c1-5395-43d8-8a0a-79396d627f66  water bottles         4
151  152  f20bbd93-1312-4f70-b257-654056412ec5         apples         1
152  153  f20bbd93-1312-4f70-b257-654056412ec5          candy         4
153  154  f20bbd93-1312-4f70-b257-654056412ec5        incense         3

                                   origin  \
0    d5566b15-b071-4acf-8e8e-c98433083b2d
1    c42f0de8-b4f0-4167-abd1-ae79e5e18eea
2    b145f396-de9b-42f1-9cc9-f5b52c3a941c
3    f4372224-759f-43b3-bc83-ca6106bba1af
4    49d0edae-9091-41bb-a08d-ab1c66bd08d5
..                                    ...
149  ee67c3b0-aa89-4b3b-8bbc-9d70695c132b
150  ee67c3b0-aa89-4b3b-8bbc-9d70695c132b
151  abc09fec-2fa0-48f6-b7c4-913620785520
152  abc09fec-2fa0-48f6-b7c4-913620785520
153  abc09fec-2fa0-48f6-b7c4-913620785520

                              destination on_time  \
0    50d33715-4c77-4dd9-8b9d-ff1ca372a2a2       1
1    172eb8f3-1033-4fb6-b66b-d0df09df3161       1
2    65e4544d-42ae-4751-9580-bdcb90e5fcda       1
3    745bee4e-710c-4538-8df1-5c146e1092a6       1
4    425b7a1a-b744-4c6b-898e-d424dd8cf18e       0
..                                    ...     ...
149  fa0ce0bb-b0d8-469d-8d42-e1153fc48272       0
150  fa0ce0bb-b0d8-469d-8d42-e1153fc48272       0
151  52479603-9957-4e4b-91eb-337c358d1755       1
152  52479603-9957-4e4b-91eb-337c358d1755       1
153  52479603-9957-4e4b-91eb-337c358d1755       1

                        driver_identifier
0    d8da0460-cf39-4f38-9fff-6c9b4e344d8a
1    293ccaec-6592-4f04-aae5-3e238fe62614
2    80988f09-91a3-4e1b-8e69-13551c53f318
3    5f79b402-655f-4d8e-8ff3-5ef05870e0ad
4    58beb5d3-98f8-4077-a964-1f04f7cb11e5
..                                    ...
149  4159e22a-d107-42e6-ba56-f9b65ad8df08
```

```
150  4159e22a-d107-42e6-ba56-f9b65ad8df08
```

Approach 2

Table 1: Create a table specifically for spreadsheet_0 and insert its data directly. Table 2: Create a second table to store the combined data from spreadsheet_1 and spreadsheet_2. This table will include the relevant columns from both spreadsheets, such as the shipment_identifier, product, quantity, origin, destination, and on_time

```
import pandas as pd
import sqlite3
import logging

# Configure logging
logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(levelname)s - %(message)s')

# File paths to CSV files
csv_file_0 = '/Users/shamalachandrappa/Desktop/Git Hub Push Everyday/Walmart Stock Price from 1972-2022/forage-walmart-task-4/da
csv_file_1 = '/Users/shamalachandrappa/Desktop/Git Hub Push Everyday/Walmart Stock Price from 1972-2022/forage-walmart-task-4/da
csv_file_2 = '/Users/shamalachandrappa/Desktop/Git Hub Push Everyday/Walmart Stock Price from 1972-2022/forage-walmart-task-4/da

# Function to create the SQLite connection
def create_connection(db_file):
    try:
        conn = sqlite3.connect(db_file)
        logging.info("Connected to SQLite database.")
        return conn
    except sqlite3.Error as e:
        logging.error(f"Error connecting to SQLite database: {e}")
        return None

# Function to drop and create the table for shipping_data_0
def setup_table_1(conn):
    try:
        cursor = conn.cursor()
        cursor.execute('DROP TABLE IF EXISTS shipping_table_0')
        cursor.execute('''
        CREATE TABLE shipping_table_0 (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            product TEXT,
            quantity INTEGER,
            origin TEXT,
            destination TEXT,
            on_time BOOLEAN,
            driver_identifier TEXT
        )
        ''')
        logging.info("Table shipping_table_0 created successfully.")
    except sqlite3.Error as e:
        logging.error(f"Error setting up the table for shipping_data_0: {e}")

# Function to drop and create the table for shipping_data_1 and shipping_data_2
def setup_table_2(conn):
    try:
        cursor = conn.cursor()
        cursor.execute('DROP TABLE IF EXISTS shipping_table_1_2')
        cursor.execute('''
        CREATE TABLE shipping_table_1_2 (
            id INTEGER PRIMARY KEY AUTOINCREMENT,
            shipment_id TEXT,
            product TEXT,
            quantity INTEGER,
            origin TEXT,
            destination TEXT,
            on_time BOOLEAN,
            driver_identifier TEXT
        )
        ''')
        logging.info("Table shipping_table_1_2 created successfully.")
    except sqlite3.Error as e:
        logging.error(f"Error setting up the table for shipping_data_1 and shipping_data_2: {e}")

# Function to load CSV data into a DataFrame
def load_csv_data(file_path):
    try:
        df = pd.read_csv(file_path)
        logging.info(f"Loaded data from {file_path}.")
        return df
    except Exception as e:
        logging.error(f"Error loading CSV file {file_path}: {e}")
        return None

# Function to insert data from shipping_data_0 into its respective table
def insert_data_table_1(conn, df_0):
    try:
        # Rename columns to fit the schema for df_0
        df_0 = df_0.rename(columns={
            'origin_warehouse': 'origin',
            'destination_store': 'destination',
            'product_quantity': 'quantity'
        })

        # Convert on_time to boolean
        df_0['on_time'] = df_0['on_time'].apply(lambda x: True if x == 'Y' else False)

        # Insert data from the first CSV file directly into the database
        df_0.to_sql('shipping_table_0', conn, if_exists='append', index=False)
        logging.info("Inserted data from shipping_data_0.csv into the database.")
    except sqlite3.Error as e:
        logging.error(f"Error inserting data into shipping_table_0: {e}")

# Function to process and insert data from shipping_data_1 and shipping_data_2 into the combined table
def insert_data_table_2(conn, df_1, df_2):
    try:
        # Convert on_time to boolean
        df_1['on_time'] = df_1['on_time'].apply(lambda x: True if x == 'Y' else False)

        # Group the data in df_1 by shipment_id and product to calculate total quantity per product per shipment
        df_1_grouped = df_1.groupby(['shipment_identifier', 'product', 'on_time']).size().reset_index(name='quantity')

        # Merge the grouped data with df_2 to get origin and destination information
        merged_df = pd.merge(df_1_grouped, df_2, on='shipment_identifier', how='inner')

        # Create a new DataFrame to hold the combined data
        combined_data = []

        # Process each row in the merged DataFrame
        for _, row in merged_df.iterrows():
            combined_data.append({
                'shipment_id': row['shipment_identifier'],
                'product': row['product'],
                'quantity': row['quantity'],
                'origin': row['origin_warehouse'],
                'destination': row['destination_store'],
                'on_time': row['on_time'],
                'driver_identifier': row['driver_identifier']
            })

        # Convert the combined data into a DataFrame
        combined_df = pd.DataFrame(combined_data)
```

Double-click (or enter) to edit

```
        logging.info("Inserted processed data from shipping_data_1.csv and shipping_data_2.csv into the database.")
    except sqlite3.Error as e:
        logging.error(f"Error inserting data into shipping_table_1_2: {e}")

# Function to fetch and save data from the database into a CSV file
def fetch_and_save_data(conn, table_name, output_file, exclude_columns=None):
    try:
        query = f'SELECT * FROM {table_name}'
        df = pd.read_sql(query, conn)

        # Drop any columns that should be excluded from the output
        if exclude_columns:
            df = df.drop(columns=exclude_columns)

        df.to_csv(output_file, index=False)
        logging.info(f"Data from {table_name} fetched and saved to {output_file}.")
    except sqlite3.Error as e:
        logging.error(f"Error fetching data from {table_name}: {e}")
    except Exception as e:
        logging.error(f"Error saving data to CSV file {output_file}: {e}")

# Main function to execute the steps
def main():
```