```
import sqlite3
import pandas as pd
import csv
import os

# --- Configuration ---
DATABASE_NAME = 'company_sales.db'
OUTPUT_CSV_SQL = 'customer_item_quantities_sql.csv'
OUTPUT_CSV_PANDAS = 'customer_item_quantities_pandas.csv'
CSV_DELIMITER = ';'

# --- 1. Database Setup (for demonstration purposes) ---
def create_and_populate_db(db_name):
    """
    Creates a SQLite database and populates it with sample data
    mimicking the assignment's schema and test case.
    """
    conn = None
    try:
        conn = sqlite3.connect(db_name)
        cursor = conn.cursor()

        # Drop tables if they already exist (for clean re-runs)
        cursor.execute("DROP TABLE IF EXISTS Orders;")
        cursor.execute("DROP TABLE IF EXISTS Sales;")
        cursor.execute("DROP TABLE IF EXISTS Customer;")
        cursor.execute("DROP TABLE IF EXISTS Items;")

        # Create tables
        cursor.execute("""
            CREATE TABLE Customer (
                customer_id INTEGER PRIMARY KEY,
                age INTEGER
            );
        """)
        cursor.execute("""
            CREATE TABLE Sales (
                sales_id INTEGER PRIMARY KEY,
                customer_id INTEGER,
                FOREIGN KEY (customer_id) REFERENCES Customer(customer_id)
            );
        """)
        cursor.execute("""
            CREATE TABLE Items (
                item_id INTEGER PRIMARY KEY,
                item_name TEXT
            );
        """)
        cursor.execute("""
            CREATE TABLE Orders (
```

```
            order_id INTEGER PRIMARY KEY,
            sales_id INTEGER,
            item_id INTEGER,
            quantity INTEGER, -- Quantity can be NULL as per assignment
            FOREIGN KEY (sales_id) REFERENCES Sales(sales_id),
            FOREIGN KEY (item_id) REFERENCES Items(item_id)
        );
    """)

    # Insert sample data into Customer
    customers_data = [
        (1, 21),  # Customer 1, age 21 (in 18-35 range)
        (2, 23),  # Customer 2, age 23 (in 18-35 range)
        (3, 35),  # Customer 3, age 35 (in 18-35 range)
        (4, 17),  # Customer 4, age 17 (outside range)
        (5, 40)   # Customer 5, age 40 (outside range)
    ]
    cursor.executemany("INSERT INTO Customer (customer_id, age) VALUES
(?, ?);", customers_data)

    # Insert sample data into Items
    items_data = [
        (101, 'x'),
        (102, 'y'),
        (103, 'z')
    ]
    cursor.executemany("INSERT INTO Items (item_id, item_name) VALUES (?,
?);", items_data)

    # Insert sample data into Sales and Orders
    # Customer 1 bought Item X on multiple occasions, totaling 10 for
Item X only
    cursor.execute("INSERT INTO Sales (sales_id, customer_id) VALUES
(1001, 1);")
    cursor.execute("INSERT INTO Orders (order_id, sales_id, item_id,
quantity) VALUES (1, 1001, 101, 5);")
    cursor.execute("INSERT INTO Orders (order_id, sales_id, item_id,
quantity) VALUES (2, 1001, 101, 5);")
    cursor.execute("INSERT INTO Orders (order_id, sales_id, item_id,
quantity) VALUES (3, 1001, 102, NULL);") # Item Y not bought
    cursor.execute("INSERT INTO Orders (order_id, sales_id, item_id,
quantity) VALUES (4, 1001, 103, NULL);") # Item Z not bought

    # Customer 2 bought one of each item only once, totaling 1 each Item
    cursor.execute("INSERT INTO Sales (sales_id, customer_id) VALUES
(1002, 2);")
    cursor.execute("INSERT INTO Orders (order_id, sales_id, item_id,
quantity) VALUES (5, 1002, 101, 1);")
    cursor.execute("INSERT INTO Orders (order_id, sales_id, item_id,
quantity) VALUES (6, 1002, 102, 1);")
    cursor.execute("INSERT INTO Orders (order_id, sales_id, item_id,
quantity) VALUES (7, 1002, 103, 1);")

    # Customer 3 bought Item Z on two occasions, totaling 2 for Item Z
only
```

```
        cursor.execute("INSERT INTO Sales (sales_id, customer_id) VALUES
(1003, 3);")
        cursor.execute("INSERT INTO Orders (order_id, sales_id, item_id,
quantity) VALUES (8, 1003, 101, NULL);") # Item X not bought
        cursor.execute("INSERT INTO Orders (order_id, sales_id, item_id,
quantity) VALUES (9, 1003, 102, NULL);") # Item Y not bought
        cursor.execute("INSERT INTO Orders (order_id, sales_id, item_id,
quantity) VALUES (10, 1003, 103, 1);")
        cursor.execute("INSERT INTO Orders (order_id, sales_id, item_id,
quantity) VALUES (11, 1003, 103, 1);")

        # Customer 4 (age 17) - should not appear in results
        cursor.execute("INSERT INTO Sales (sales_id, customer_id) VALUES
(1004, 4);")
        cursor.execute("INSERT INTO Orders (order_id, sales_id, item_id,
quantity) VALUES (12, 1004, 101, 2);")

        conn.commit()
        print(f"Database '{db_name}' created and populated successfully.")
    except sqlite3.Error as e:
        print(f"Database error during setup: {e}")
    finally:
        if conn:
            conn.close()

# --- 2. Data Extraction - Solution 1: Pure SQL ---
def extract_data_sql(db_name, output_csv):
    """
    Extracts data using a pure SQL query and saves it to a CSV file.
    """
    conn = None
    try:
        conn = sqlite3.connect(db_name)
        cursor = conn.cursor()

        sql_query = """
            SELECT
                C.customer_id AS Customer,
                C.age AS Age,
                I.item_name AS Item,
                SUM(O.quantity) AS Quantity
            FROM
                Customer AS C
            JOIN
                Sales AS S ON C.customer_id = S.customer_id
            JOIN
                Orders AS O ON S.sales_id = O.sales_id
            JOIN
                Items AS I ON O.item_id = I.item_id
            WHERE
                C.age BETWEEN 18 AND 35
                AND O.quantity IS NOT NULL -- Exclude NULL quantities (items
not bought)
            GROUP BY
                C.customer_id, C.age, I.item_name
```

```
            HAVING
                SUM(O.quantity) > 0; -- Exclude items with total quantity 0
        """
        cursor.execute(sql_query)
        results = cursor.fetchall()

        # Get column headers
        headers = [description[0] for description in cursor.description]

        # Write to CSV
        with open(output_csv, 'w', newline='', encoding='utf-8') as f:
            writer = csv.writer(f, delimiter=CSV_DELIMITER)
            writer.writerow(headers)
            for row in results:
                # Ensure quantity is integer (no decimal points)
                # sqlite3.Row objects behave like tuples, so direct indexing
works
                formatted_row = list(row)
                if isinstance(formatted_row[3], (float, int)): # Assuming
quantity is the 4th column (index 3)
                    formatted_row[3] = int(formatted_row[3])
                writer.writerow(formatted_row)

        print(f"SQL query results saved to '{output_csv}'")

    except sqlite3.Error as e:
        print(f"Database error during SQL extraction: {e}")
    except IOError as e:
        print(f"File I/O error during SQL extraction: {e}")
    finally:
        if conn:
            conn.close()

# --- 3. Data Extraction - Solution 2: Pandas ---
def extract_data_pandas(db_name, output_csv):
    """
    Extracts data using Pandas DataFrames and saves it to a CSV file.
    """
    conn = None
    try:
        conn = sqlite3.connect(db_name)

        # Load data into Pandas DataFrames
        df_customer = pd.read_sql_query("SELECT * FROM Customer;", conn)
        df_sales = pd.read_sql_query("SELECT * FROM Sales;", conn)
        df_orders = pd.read_sql_query("SELECT * FROM Orders;", conn)
        df_items = pd.read_sql_query("SELECT * FROM Items;", conn)

        # Merge DataFrames
        df_merged = pd.merge(df_customer, df_sales, on='customer_id')
        df_merged = pd.merge(df_merged, df_orders, on='sales_id')
        df_merged = pd.merge(df_merged, df_items, on='item_id')

        # Filter by age (18-35)
        df_filtered_age = df_merged[(df_merged['age'] >= 18) &
```

```
(df_merged['age'] <= 35)]

        # Filter out NULL quantities (items not bought)
        df_filtered_quantity =
df_filtered_age[df_filtered_age['quantity'].notna()]

        # Group by customer_id, age, item_name and sum quantities
        df_grouped = df_filtered_quantity.groupby(['customer_id', 'age',
'item_name'])['quantity'].sum().reset_index()

        # Filter out items with total quantity 0
        df_final = df_grouped[df_grouped['quantity'] > 0]

        # Ensure quantities are integers (no decimal points)
        df_final['quantity'] = df_final['quantity'].astype(int)

        # Rename columns to match desired output
        df_final.rename(columns={
            'customer_id': 'Customer',
            'age': 'Age',
            'item_name': 'Item',
            'quantity': 'Quantity'
        }, inplace=True)

        # Save to CSV
        df_final.to_csv(output_csv, sep=CSV_DELIMITER, index=False,
encoding='utf-8')

        print(f"Pandas query results saved to '{output_csv}'")

    except sqlite3.Error as e:
        print(f"Database error during Pandas extraction: {e}")
    except Exception as e:
        print(f"An error occurred during Pandas extraction: {e}")
    finally:
        if conn:
            conn.close()

# --- Main Execution ---
if __name__ == "__main__":
    # Clean up previous output files if they exist
    if os.path.exists(OUTPUT_CSV_SQL):
        os.remove(OUTPUT_CSV_SQL)
    if os.path.exists(OUTPUT_CSV_PANDAS):
        os.remove(OUTPUT_CSV_PANDAS)
    if os.path.exists(DATABASE_NAME):
        os.remove(DATABASE_NAME)

    # 1. Create and populate the dummy database
    create_and_populate_db(DATABASE_NAME)

    # 2. Extract data using Pure SQL solution
    extract_data_sql(DATABASE_NAME, OUTPUT_CSV_SQL)

    # 3. Extract data using Pandas solution
```

```
extract_data_pandas(DATABASE_NAME, OUTPUT_CSV_PANDAS)

print("\n--- Script Finished ---")
print(f"Check '{OUTPUT_CSV_SQL}' and '{OUTPUT_CSV_PANDAS}' for results.")
```