Complete Documentation: Duplicate Emails Problem

Problem Overview:

You are provided with a table called Person that contains two columns:

- **id:** A unique integer representing the identifier for each record (the primary key).
- **email:** A string representing the email address for each record. The emails do not contain uppercase letters, and it is guaranteed that the email field is not NULL.

The task is to find and report the emails that appear more than once in the table, i.e., the duplicate emails. The result should contain only the duplicate email addresses in a single column named Email.

Key Points:

1. **Input Structure:**

- The input is a DataFrame that mimics the structure of the Person table.
- The DataFrame contains two columns: id and email.
- Each row represents one email address, and emails may be repeated.

2. **Output Structure:**

- The output should be a DataFrame with a single column named Email.
- This column will contain only the email addresses that appear more than once in the input DataFrame.

3. Approach:

- *The problem can be solved in the following steps:*
  - ***Step 1:** Grouping by Email:*
    - ✓ The first step is to group the data by the email column to count how many times each email appears in the table.
  - ***Step 2:** Counting Occurrences:*
    - ✓ After grouping the data, count the occurrences of each email. This allows you to identify emails that appear more than once.
  - ***Step 3:** Filtering Duplicates:*
    - ✓ Once you have the count of occurrences for each email, filter the data to keep only those emails that have appeared more than once.
  - ***Step 4:** Formatting the Output:*
    - ✓ The final step is to return the filtered list of emails in a DataFrame. The column in the output should be renamed to Email, following the required format.

4. Edge Cases to Consider:

- ***No Duplicates:***
  - If there are no duplicate emails, the result should be an empty DataFrame with a column named Email.
- ***All Unique Emails:***
  - In the case where all emails are unique, the output will also be an empty DataFrame since there are no duplicates to report.

- ***Multiple Duplicates:***
  - If multiple email addresses have duplicates, the output should contain all such addresses. Each duplicate email should only appear once in the output, even if it appears multiple times in the input.

Example 1:

Input:

The Person table is represented as:

id	email
1	a@b.com
2	c@d.com
3	a@b.com

Expected Output:

Email
a@b.com

Explanation:

- The email a@b.com appears twice, so it is identified as a duplicate.

Constraints:

- **Primary Key:**
  - The id column is the primary key, meaning each id is unique. However, the email column is not unique, which allows for duplicate entries.

- **Email Characteristics:**
  - The email addresses are guaranteed to contain only lowercase letters and will not be NULL. This simplifies validation checks during processing.

Efficiency Considerations:

- **Time Complexity:**
  - The primary operation involves grouping and counting, which typically runs in O(n) time, where n is the number of rows in the input table. This is efficient enough for typical data sizes.

- **Space Complexity:**
  - The space complexity is also O(n) since you need to store the group counts and the final filtered list of duplicates. However, this space usage is manageable within the constraints of most systems.

Summary:

- The task requires identifying duplicate emails from the Person table and reporting them in a specific format. By grouping emails and filtering those that appear more than once, the problem can be efficiently solved with minimal complexity. The function should handle all edge cases, including tables with no duplicates or entirely unique emails.