Boundary value and Equivalence class analysis

There are 3 values that are provided as input to the program:
input csv filename 1, input csv filename 2, and a string denoting the unique combinations
We will explicitly list out all the boundary values and equivalence values for the input filenames, file contents, unique combination strings, however I will note that for all the equivalence classes for each input, we should also consider the permutations of all the equivalence classes for each of the inputs. I won't explicitly state all the permutations cause there's too many, but I might mention a few instances where the combination itself is unique.

Filename input

1. Boundary values
    a. NULL
    b. Empty filename (“”)
2. Equivalence classes
    a. Invalid filename (contains invalid characters like $)
    b. Filename points to non existent file
    c. Filename actually points to a directory instead
    d. Filename input points to file without read permissions
    e. Filename input has no extension
    f. Filename input has trailing whitespace
    g. Filename input has leading whitespace

File contents

1. Boundary values
    a. File is empty
2. Equivalence classes
    a. File contains null character “\0’ at various (>=1) places in the program
    b. File has only 1 row that has headers only
    c. File has more than 1 row including headers only
    d. File has no header row
    e. File contains blank whitespace lines inside between proper rows
    f. File is just whitespace
    g. File has inconsistent number of columns and rows within the file
    h. File has no headers

i. Header columns are not unique (duplicate columns)
j. Some of header column names have white space around them
   E.g. “ Mactones , Ketones” vs “Mactones,Ketones”
k. None of the header column names what whitespace around them
l. Column names have space in them e.g. “fish size, fish age”
m. Some of the column values for some rows (>=1) are wrapped in whitespace in the raw file content e.g. “123, 14, 43, 3”
n. None of the column values for some rows (>=1) are wrapped in whitespace in the raw file content
o. File has N > 1 columns and M > 1 rows and all the rows have the same number of columns
p. File has 1 row and 1 column, excluding the header row
q. File has 1 row and M > 1 columns, excluding the header row
r. File has N > 1 rows and 1 column, excluding the header row
s. File has N > 1 rows and a header row but no other rows
t. Header column names contain null character e.g. “ errors \0 contained, tests written”
u. Header column names contain other whitespace characters e.g. “ errors \t contained, tests written”

Unique combination raw input (comma delimited string)

1. Boundary values
   a. NULL
   b. Empty string
   c. String that consists of whitespace only
2. Equivalence classes
   a. One column value only (no comma)
   b. Double consecutive commas (price ,, size)
   c. N consecutive commas (N > 2)
   d. Duplicate column names
   e. Some of typed column names have white space around them
      E.g. “ Potatoness , Ketones” vs “Potatoness,Ketones”
   f. None of the typed column names what whitespace around them
   g. Column names have space in them e.g. “fish size, fish age”
   h. column names contain null character e.g. “ errors \0 contained, tests written”
   i. column names contain other whitespace characters e.g. “ errors \t\n contained, tests written”

Cross file input / unique combination string permutation considerations

1. Files do not having matching number of columns
2. Column headers have the same set of values but column order is different across the input csv files
3. Columns in unique combination string are the same as the columns in the csv files
4. Columns in unique combination string are not the same as the columns in the csv files
5. Number of columns in unique combination string are the same as the number of columns in the csv files
6. Number of columns in unique combination string are not the same as the number of columns in the csv files
7. Some of the column values for some rows (>=1) are wrapped in whitespace in the raw file content e.g. “123, 14, 43, 3” , AND there exists a row somewhere in the the other file with fundamentally the same values but that has different whitespace around the values e.g. “ 123, 14 , 43 , 3”
8. FILES CONTENTS WHERE THE FOLLOWING DOES NOT EXIST: Some of the column values for some rows (>=1) are wrapped in whitespace in the raw file content e.g. “123, 14, 43, 3” , AND there exists a row somewhere in the the other file with fundamentally the same values but that has different whitespace around the values e.g. “ 123, 14 , 43 , 3”
9. Some of the column values for some rows (>=1) are wrapped in whitespace in the raw file content e.g. “123, 14, 43, 3” , AND there exists a row somewhere in the the other file with fundamentally the same values but that has no whitespace around the values e.g. “123,14,43,3”

   Lol you may be wondering what's the difference between this and the one where both files have different whitespace, but it's possible the code dies when it tries to remove whitespace from a row without whitespace to start with maybe.

10. FILES CONTENTS WHERE THE FOLLOWING DOES NOT EXIST: Some of the column values for some rows (>=1) are wrapped in whitespace in the raw file content e.g. “123, 14, 43, 3” , AND there exists a row somewhere in the the other file with fundamentally the same values but that has no whitespace around the values e.g. “123,14,43,3”