# DataEng S22: Data Validation Activity

High quality data is crucial for any data project. This week you'll gain experience with validating a real data set.

**Submit**: Make a copy of this document and use it to record your results. Store a PDF copy of the document in your git repository along with any needed code before submitting using the in-class activity submission form.

## Initial Discussion Question - Discuss the following question among your working group members at the beginning of the week and place your responses in this space. Or, if you have no such experience with invalid data then indicate this in the space below.

*Have you ever worked with a set of data that included errors? Describe the situation, including how you discovered the errors and what you did about them.*

Response 1: Jeremy Hamilton
Working with a database where the column can be null, trying to differential the empty data and null data.

Response 2: Leshi Chen
I have only experience with SQL. If I try to input an empty value into the primary key column, the command will fail.

Response 3: Morgan Courvoisier
Dont's particularly have cleaned data issues. The company Morgan is working on has a validation check by a third corporation handled.

Response 4:

The data set for this week is a listing of all Oregon automobile crashes on the Mt. Hood Hwy (Highway 26) during 2019. This data is provided by the Oregon Department of Transportation and is part of a larger data set that is often utilized for studies of roads, traffic and safety.

Here is the available documentation for this data: description of columns, Oregon Crash Data Coding Manual

Data validation is usually an iterative three-step process.

A. Create assertions about the data
B. Write code to evaluate your assertions.

C. Run the code, analyze the results and resolve any validation errors

Repeat this ABC loop as many times as needed to fully validate your data.

## A. Create Assertions

Access the crash data, review the associated documentation of the data (ignore the data itself for now). Based on the documentation, create English language assertions for various properties of the data. No need to be exhaustive. Develop one or two assertions in each of the following categories during your first iteration through the ABC process.

1. *existence* assertions. Example: “Every crash occurred on a date”
2. *limit* assertions. Example: “Every crash occurred during year 2019”
3. *intra-record* assertions. Example: “Every crash has a unique ID”
4. Create 2+ *inter-record check* assertions. Example: “Every vehicle listed in the crash data was part of a known crash”
5. Create 2+ *summary* assertions. Example: “There were thousands of crashes but not millions”
6. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”

These are just examples. You may use these examples, but you should also create new ones of your own.

1. Existence Assertion: Every crash occurred on a date.
2. Limite Assertion: Every crash occurred during the year 2019.
3. Intra-record Assertion: Every crash has a unique ID.
4. Inter-record Assertion 1: Every vehicle ID listed in the crash data was part of a known crash.
   Inter-record Assertion 2: Every participant ID listed in the crash data was part of a known crash.
5. Summary Assertion 1: There were thousands of crashes but not millions.
   Summary Assertion 2: There were thousands of vehicles involved in crashes but not millions.
6. Statistical Assertion 1: Crashes are evenly/uniformly distributed throughout the months of the year.
   Statistical Assertion 2: The number of vehicles crashes are evenly/uniformly distributed throughout the months of the year.

## B. Validate the Assertions

1. Study the data in an editor or browser. Study it carefully, this data set is non-intuitive!.
2. Write python code to read in the test data. You are free to write your code any way you like, but we suggest that you use pandas' methods for reading csv files into a pandas Dataframe.
3. Write python code to validate each of the assertions that you created in part A. The pandas package eases the task of creating data validation code.
4. If needed, update your assertions or create new assertions based on your analysis of the data.

## C. Run Your Code and Analyze the Results

In this space, list any assertion violations that you encountered:

- I set up the different conditions to help me validate the data, so I don't think I have met particular assertion violations at this point.
- I did meet some trouble with some calculations of the data, I revised the assumptions/assertions technique.

For each assertion violation, describe how to resolve the violation. Options might include:

- revise assumptions/assertions
- discard the violating row(s)
- Ignore
- add missing values
- Interpolate
- use defaults
- abandon the project because the data has too many problems and is unusable

No need to write code to resolve the violations at this point, you will do that in step E.

## D. Learn and Iterate

The process of validating data usually gives us a better understanding of any data set. What have you learned about the data set that you did not know at the beginning of the current ABC iteration?

I have learned the data set could help us to analyze the frequency of crashes that happened on Oregon Hwy 26 throughout the year 2019. Not just that, by analyzing the data set in different ways we can get to know different information/statistics to us.

Next, iterate through the process again by going back through steps A, B and C at least one more time.

## E. Resolve the Violations and Transform the Data

For each assertion violation write python code to resolve the violation according to your entry in the “how to resolve” section above.

Output the validated/transformed data to new files. There is no need to keep the same, awkward, single file format for the data. Consider outputting three files containing information about (respectively) crashes, vehicles and participants.

I have output the assertion violation to an external file named errors, and not just assertion violations and it also help me to test the coding itself. What did was keep the single file format for the data, I will only separate them when I get to a certain condition such as when I want to test if a crash has to have a date involved. I will set the record type to three and help me verify if this crash has a date or not.