

# DataEng: Data Validation Activity

## [In-class Activity Submission Form](#)

Ted Herring

High quality data is crucial for any data project. This week you'll gain some experience and knowledge of analyzing data sets for quality.

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.

- First (part A) you develop assertions about your data as a way to make your assumptions explicit.
- Second (part B) you write code to evaluate the assertions and test the assumptions. This helps you to refine your existing assertions (part C) before starting the whole process over again by creating new assertions (part A again).

## 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 for this assignment, two or more assertions in each category are enough.

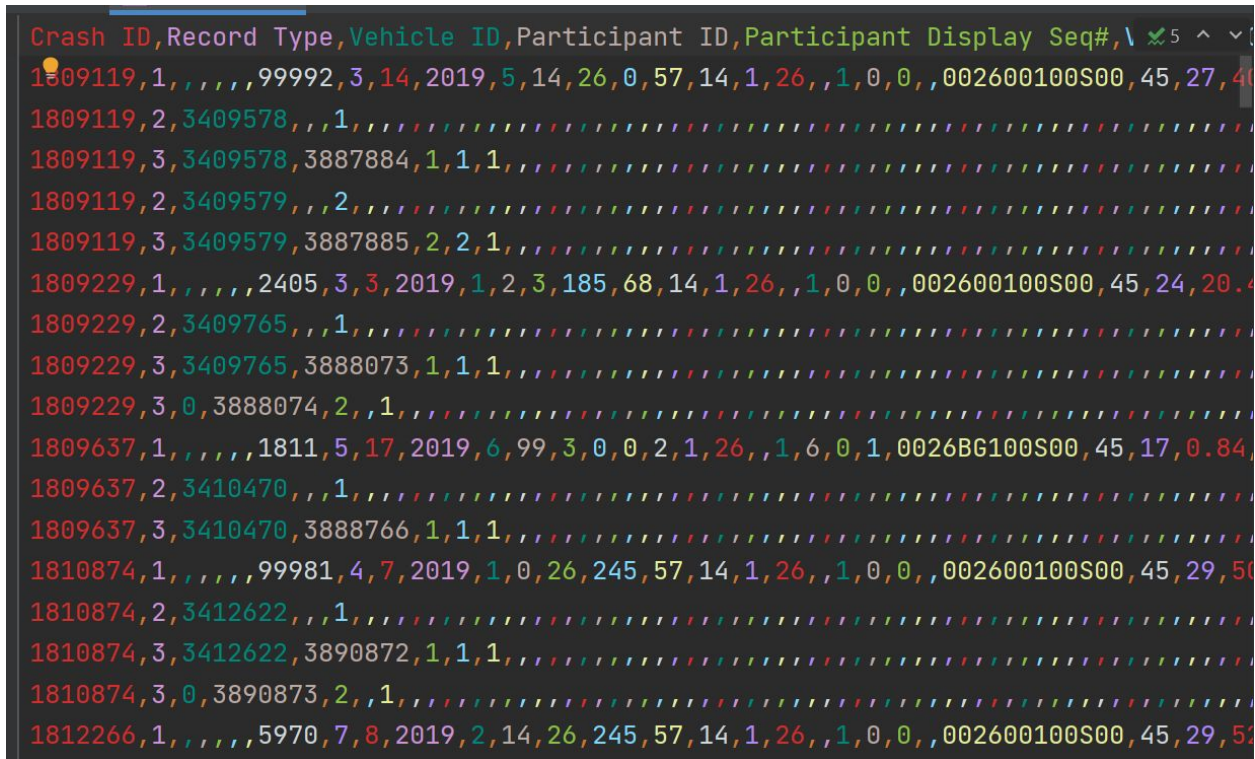
1. Create 2+ *existence* assertions. Example, “Every record has a date field”.
  - DataFieldsAndFormats.xlsx
    - i. Decode\_crash
      1. Database field names are hard to read, we’ll want to strongly tie them to their label
      2. Datatypes are varied with defined lengths
      3. Column *format* gives a numeric value, unsure what that is
      4. Comments exist for a number of rows
    - ii. Decode\_vehicle
      1. No *format* column
    - iii. Decode\_participant
      1. No *format column*
2. Create 2+ *limit* assertions. The values of most numeric fields should fall within a valid range. Example: “the date field should be between 1/1/2019 and 12/31/2019 inclusive”
  - Day of week code looks to be limited to a length of 1 char. I wonder how that will work for tues vs thurs sat vs. sunday
  - Intersection sequence number is limited to a *Auto* I dunno what that means
  - The crash type short description is only 10 chars, that’s pretty short
    - i. Ok so after working with the data for all of Thursday, I’m just going round in circles and I can’t figure out what to assert. With the multiple record types I’m very confused. So as for assertions about the data, I’m still working on finding an entry point. I feel like Tuesday I should go back to the documentation to see what we have to go off of there. At this point I may be able to isolate record type 1 from the other two and let them just hang out for a bit.
    - ii. It’s a weird data set to be sure. So still unsure how I want to organize it. I’m gonna have to rethink my approach, once I’m confident about an approach I think I can move nicely through the rest of this assignment, but as for Thursday I’ve done a lot of meaningful thinking lol
    - iii.
3. Create 2+ *intra-record check* assertions.

- Check vehicle crash ID matches actual crash ID and cross reference with vehicle count
  - Check participant crash ID matches actual crash ID and cross reference with participant count
  - Check for duplicates for unique fields
4. Create 2+ *inter-record check* assertions.
- Check BAC volume is between 80 (.080 legally intoxicated) and 400(.400 dead)
  - Check Vehicle coded sequence number corresponds to Code Manual - "Vehicle Number"

5. Create 2+ *summary* assertions. Example: “every crash has a unique ID”
  - Crashes vehicles and participants should all have crash IDs
  - Crashes vehicles and participants should all have vehicle IDs
6. Create 2+ referential integrity assertions. Example “every crash participant has a Crash ID of a known crash”
  - Some of those are made above
  -
7. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the year.”
  - Day of week regularity
  - Time of day regularity
  - Location regularity
  - Type of car involved
  - Severity distribution

## B. Validate the Assertions

1. Now study the data in an editor or browser. If you are anything like me you will be surprised with what you find. The Oregon DOT made a mess with their data!
  - o Yeah no kidding, looks at all the pretty commas in pycharm, lots of missing data!



```
Crash ID,Record Type,Vehicle ID,Participant ID,Participant Display Seq#,
1809119,1,,,,,99992,3,14,2019,5,14,26,0,57,14,1,26,,1,0,0,,002600100S00,45,27,40
1809119,2,3409578,,1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1809119,3,3409578,3887884,1,1,1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1809119,2,3409579,,2,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1809119,3,3409579,3887885,2,2,1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1809229,1,,,,,2405,3,3,2019,1,2,3,185,68,14,1,26,,1,0,0,,002600100S00,45,24,20.4
1809229,2,3409765,,1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1809229,3,3409765,3888073,1,1,1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1809229,3,0,3888074,2,,1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1809637,1,,,,,1811,5,17,2019,6,99,3,0,0,2,1,26,,1,6,0,1,0026BG100S00,45,17,0.84,
1809637,2,3410470,,1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1809637,3,3410470,3888766,1,1,1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1810874,1,,,,,99981,4,7,2019,1,0,26,245,57,14,1,26,,1,0,0,,002600100S00,45,29,50
1810874,2,3412622,,1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1810874,3,3412622,3890872,1,1,1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1810874,3,0,3890873,2,,1,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,,
1812266,1,,,,,5970,7,8,2019,2,14,26,245,57,14,1,26,,1,0,0,,002600100S00,45,29,50
```

2. Write python code to read in the test data and parse it into python data structures. You can write your code any way you like, but we suggest that you use pandas' methods for reading csv files into a pandas Dataframe
  - o Ugh, I got stuck here forever and I may have just been on a total tangent. I really got hung up on the type of data structure to use, given that each crash ID has multiple entries. I ended up with a dictionary (indexed by crash id) with a list of all the details contained therein, each indexed sequentially. We'll see how this works out
  - o Ah, now I see the record type, I need to clean this up to get data structures for each record type
3. Write python code to validate each of the assertions that you created in part A. Again, pandas makes it easy to create and execute assertion validation code.
  - o Ok so this is kinda ugly, I've now broken the data into 3 data structures based on the record types. From here I'll sanitize the data, removing unneeded columns, and making my assertions. I think with data that's as messy as this I'm going to have to work through some explicit and clear data manipulation which will be

fairly inefficient, then once I know how I can clean it all up, go back and streamline my approach.

4. If you are like me you'll find that some of your assertions don't make sense once you actually understand the structure of the data. So go back and change your assertions if needed to make them sensible.
  - This is for sure, it really looks like all my struggling is still ongoing, I fell down a rabbit hole, though Tuesdays lecture helped get me back on the right path. Now I'm trying to find how to access the data per row to iterate over the data to make my assertions. I feel like I need to spend a few hours with pandas to get more familiar with the data structures involved. It is cool though, now having used it in the SRE class and now here, I can see how it's useful. I'm actually designing a project at work which I will be using it too so all this struggling does have a point, so that's good.
  - As for data assertions, I'll check to see if dates are in range for sure, the rest I'm not sure where the others should lie. I'll go back to my originals and reconsider at this point. Though that's after I find a way to actually make the assertion.
  - Ok I think I got it, if we iterate with a python for loop over the crashes DF, we can then match a sting like 'Crash Day' to the iterator, then we can iterate over the DF using the outerloop iterator as our index. Perfect!
5. Run your code and note any assertion violations. List the violations here.
  - Found 4 violations to checking the crash hour, in 4 rows the crash hour is listed as 99, we'll have to fix that!

## C. Evaluate the Violations

For any assertion violations found in part B, describe how you might resolve the violation. Options might include "revise assumptions/assertions", "discard the violating row(s)", "ignore", "add missing values", "interpolate", "use defaults", etc.

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

If you chose to "revise assumptions/assertions" for any of the violations, then briefly explain how you would revise your assertions based on what you learned.

## 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 ABCD iteration?

Next, iterate through the process again by going back to Step A. Add more assertions in each of the categories before moving to steps B and C again. Go through the full loop twice before moving to step E.

## E. Resolve the Violations

For each assertion violation found during the two loops of the process, write python code to resolve the assertions. This might include dropping rows, dropping columns, adding default values, modifying values or other operations depending on the nature of the violation.

Note that I realize that this data set is somewhat awkward and that it might be best to “resolve the violations” by restructuring the data into proper tables. However, for this week, I ask that you keep the data in its current overall structure. Later (next week) we will have a chance to separate vehicle data and participant data properly.

## E. Retest

After modifying the dataset/stream to resolve the assertion violations you should have produced a new set of data. Run this data through your validation code (Step B) to make sure that it validates cleanly.

Submit: [In-class Activity Submission Form](#)