DataEng S24: Data Validation Activity

High quality data is crucial for any data project. This week you'll gain experience with validating a real data set provided by the Oregon Department of Transportation.

Due: this Friday at 10pm PT
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.

A. [MUST] Initial Discussion Question

Discuss the following question among your working group members at the beginning of the week and place your own response(s) 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: No. I havent encountered any dataset with errors.

Background

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 multi-step process.

B. Create assertions about the data
C. Write code to evaluate your assertions.
D. Run the code, analyze the results
E. Write code to transform the data and resolve any validation errors

B. [MUST] 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: “If a crash record has a latitude coordinate then it should also have a longitude coordinate”
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.

Answer:

1. Existence assertions:
-> Every crash has a crash id associated with it.

2. Limit Assertions:
-> Every crash month must be a valid month number (01-12).

3. Intra-Record assertion:
-> If the crash has a city code it should have a urban area code associated with it.
-> Number of turning Legs must be numeric when Road Character is Intersection.

4. Inter-record assertions:
-> If 'Participant Error 1 Code', 'Participant Error 2 Code', or 'Participant Error 3 Code' is not 'None', then 'Participant Cause 1 Code', 'Participant Cause 2 Code', or 'Participant Cause 3 Code' should not be 'None' for each participant involved in a crash.
-> If 'Vehicle Hit & Run Flag' is 1 for any vehicle involved in a crash, then 'Participant Hit & Run Flag' should be 1 for at least one participant involved in the crash.

5.Summary Assertions:
-> Every crash occurs on a Highway Number 26.
-> All the crashes involved have involved speed flag as US or Oregon.

6. Statistical Assertions:
-> Alcohol involved crashes are very less when compared to crashes which didn't have alcohol involved.
-> Majority of the crashes have the total number of vehicle count involved in the crash as 1.

C. [MUST] 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.

D. [MUST] Run Your Code and Analyze the Results

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

- For Limit Assertions, record type two and record type three doesn't have a valid month number.
- For Summary Assertion, 'every crash occurs on highway number 26', the assertion is violated.
- For Summary Assertion, 'All the crashes involved have involved speed flag as US or Oregon.', few records violates the assertion.
- For Statistical Assertion,' Majority of the crashes have the total number of vehicle count involved in the crash as 1.', few records violates the assertion.
- For the inter-record transition,' If 'Participant Error 1 Code', 'Participant Error 2 Code', or 'Participant Error 3 Code' is not 'None', then 'Participant Cause 1 Code', 'Participant Cause 2 Code', or 'Participant Cause 3 Code' should not be 'None' for each participant involved in a crash., few records violates the assertion.
- For inter-record transition,' If 'Vehicle Hit & Run Flag' is 1 for any vehicle involved in a crash, then 'Participant Hit & Run Flag' should be 1 for at least one participant involved in the crash.',few record violates the assertion.

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.

Answer:

- ➔ Limit assertion violation was resolved by changing the assertion to check the limit only for record type 1.
- ➔ For inter-record violation for the assertion,' If 'Participant Error 1 Code', 'Participant Error 2 Code', or 'Participant Error 3 Code' is not 'None', then 'Participant Cause 1 Code', 'Participant Cause 2 Code', or 'Participant Cause 3 Code' should not be 'None' for each participant involved in a crash.', I have ignored the rows which violated the assertions.
- ➔ For inter-record transition,' If 'Vehicle Hit & Run Flag' is 1 for any vehicle involved in a crash, then 'Participant Hit & Run Flag' should be 1 for at least one participant involved in the crash.',few record violates the assertion, I have modified the values i.e I have added a value of '1' to Participant Hit & Run Flag wherever it was missing.
- ➔ For the other assertion violations, I have ignored the rows with null values.

E. [SHOULD] 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.

- ➔ Python code is in GitHub.

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 divided the original file into three different csv files based on record type.

F. [ASPIRE] 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?

Answer: The given dataset is very inconsistent. There are a lot of missing values. Upon reading the documentation for the dataset, I found many violations. Also, there is no proper documentation for the added values for many attributes. There are many columns present which have no value in it. This has made the dataset very inconsistent.

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

➔ I iterated through the dataset once and all the assertions passed.