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:

- **Trae** - "My first time working with seperate tables in order to combine them into one was when I worked with different recipe and food tables in order to make a cohesive table with similiar nutrients and calories. Unfortunately the types of measurements, some of the descriptions of the food and links to photos in other tables made it so there were entire columns of errors. In order to make similar tables I had to limit the similar nutrients in each food and dish to the most minimal I could so that the columns that caused massive difference weren't a problem."
- **Jonathan** - "I have not really had too much experience with data. I would say that the time I did I was working on a class project for my database management class and some of the data was very obviously invalid. Since there was so much data scrapping the sets with errors was irrelevant and also quick. Since there was so much data, getting rid of the erros flat out instead of fixing them did not lead to very drastic results."
- **Chase** - "Working to validate, update, and add to a GeoJSON dataset recording geographic information on Elementary, Middle, and High Schools in Washington State- I encountered a lot of erroneous or non-existent addresses. I also found that any small alteration in the lat/long coordinates could easily cause the point to be unrepresentative of the true location. This made verification through ArcGIS with the proper coordinate system tedious but very necessary."
- **Will** - "Mine is that I worked with a real-estate company trying to predict a homeowner's likelihood of foreclosing in a year from their date of purchase. The data came from

various brokers and therefore various formats (api, graphQL, csv dump) and parsing through the concatenating the information was hard. One broker in particular had important values where both "NaN" and "0" meant "none" while anything higher meant those values. For instance one was "number_of_credit_cards" and some values meant "NaN" (like they don't know) and some meant "0" they didn't have any cards. That was annoying. Also we had street addresses and various formats which were also annoying when trying to connect information from various brokers."

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.

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:

- My assertion number 5. There are some serious inconsistencies with the values of the 'Crash Severity' values when comparing the value counts of the data versus what should be there according to the data dictionary.
  - This I would go back to the data brokers and ask them what’s up with the discrepancy and how we should approach it.
- My assertion number 8. I would not expect there to be crashes without participants in the vehicle.
  - For this I would ignore it. I’m sure that there’s a reason for this.
- My assertion number 9. I would expect the sum of total vehicle, fatality, and injury counts should match the total persons involved count.
  - For this I would go back to the data brokers and ask them what’s up with the discrepancy and how we should approach it.
- My assertion number 11. Each month has a different number of crashes which is something I would expect that they would vary.

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.

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.

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.

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?

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