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:
Sriram:
Unreadable non-Ascii characters. Solution was to rewrite the characters. Other situations, there were cases where entries had values outside of acceptable limits. Those entries were removed.

Alex:
Within this project for DataEng, fields were marked as None when pulling from pubsub. Json module doesn't parse python None objects. Solution was to rewrite None as 'None' string.

Kirk:
Wafer/die data from Intel, a lot of die had empty values or -999 value for certain parameters. I removed those entries.

Monica:
I worked for my UG project where the data had missing values,outliers and inconsistencies which required me to use outlier detection,standardizations and validations for generating accurate segmentation which is reliable for my clustering project.

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”
   Every crash should have a vehicle ownership.

2. *limit* assertions. Example: “Every crash occurred during year 2019”
   Every serial # is at most 5 digits

3. *intra-record* assertions. Example: “If a crash record has a latitude coordinate then it should also have a longitude coordinate”
   If there’s a turning lane, there’s a median type

4. Create 2+ *inter-record check* assertions. Example: “Every vehicle listed in the crash data was part of a known crash”
   Every crash has a Crash ID.
   Every crash has a Record Type

5. Create 2+ *summary* assertions. Example: “There were thousands of crashes but not millions”

Most crashes have a participant ID.
Most Crashes have a Vehicle ID

6. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”
   Most crashes occurred between midnight and 3am.
   Most crashes occurred during the summer months

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:
Only my statistics assertions were violated. I forgot this was a highway running through Mt. Hood so it makes sense that most happened during the day, and that most were also during the winter months (October - Early-to-mid-April)

- ~~**Most crashes occurred between midnight and 3am.**~~ **VIOLATED!**
  - **Most crashes happen during the day. Fact!**
- ~~**Most crashes occurred during the summer months.**~~ **VIOLATED!**
  - **Most crashes occurred during the winter. Fact!**

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

I elected to adjust my assumptions to see the data portrayed more accurately based on the assertions. Again, it was my stats assertions that failed. For the other assertions, to accurately see if my assumptions were right, I dropped empty rows in that column as they are not helpful for analyzing the facts. For those that were empty, I would have tried to extrapolate related data surrounding the cell to get the right information.

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.