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 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. [Note: this part will be

Response 1: Alex - I can't think of any data sets that I've worked with that had errors, so this will be my first.

Response 2: Joseph - I have not worked with a set of data that included errors.

Response 3: Joey - I haven't worked with any data sets that have validation errors

Response 4: Leo - Most common problems encountered for attribution analytics at TUNE:

1. timezone issues - data crossovers thanks to daylight savings etc.
2. Different languages unicoding causing issues in urls
3. currency lock-in rate issues causing client conflicts

Probably more can't recall

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.

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:
   a. “Every crash must have a crash ID”
   b. “Each crash must have a record type”
2. *limit* assertions. Example: “Every crash occurred during year 2019”
   A. Record type cannot be above 3
   B. Crash month cannot be greater than 12
3. *intra-record* assertions. Example: “If a crash record has a latitude coordinate then it should also have a longitude coordinate”
   A. “If a crash record has at least one pedalcyclist involved in the crash, then either the total pedalist fatality count, or the Total Pedal-cyclist Non-Fatal Injury Count cannot be null. ”
   B. “Total Count of Persons Involved is greater than 0, than the fields “Total Pedestrian Count + Total Pedalcyclist Count + Total Unknown Count + Total Vehicle Occupant Count” cannot all be null”
4. Create 2+ *inter-record check* assertions. Example: “Every vehicle listed in the crash data was part of a known crash”
   A. “For every crash record with a participant ID field that is not null, there should be another crash record with the same vehicle Id but with a null participant ID“
   B. “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”
   A. “The majority of crashes occurred during daytime hours”
   B. “More crashes occur when there is rain on the road rather than when it is clear”
6. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”

A. “The distribution of crashes by road surface conditions varies, with a higher frequency of crashes on wet or icy surfaces during winter months”
B. “The amount of crashes across different counties is evenly distributed”

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:

- Each Vehicle ID should have a crash ID associated with it, but there would be one vehicle ID but with three or four crash IDs associated with it
-
-
-

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] 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?

Well for one, it is multiple datasets compiled into one CSV file. This is the purpose of the record type column, as it defines with group the data is apart of. Two, there are really only about 500 individual crashes, however, each crash has multiple Crash records associated with it, making the total amount of rows about 2000+

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

F. [ASPIRE] 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.