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: I have worked as an intern for a project in which I did data analysis. The initial steps include checking and cleaning data. So then I have come across duplicate data a lot of times.

When I deal with duplicate data, my first step is to identify and understand why the duplicates exist. Are they simply repeated entries, or are there slight variations that might need further examination? Once I pinpoint the duplicates, I decide based on their nature—either remove them if they are exact copies or merge them if they contain unique pieces of information. If I'm unsure, I flag them for a detailed review. It's crucial to address the root causes of these duplicates, such as data entry errors or issues during data consolidation, to prevent similar problems in the future. Regular checks and maintaining clear documentation of how I handle duplicates are key practices that help ensure the data remains accurate and reliable for analysis.

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.
    - Every participant record has both a Participant ID and a Crash ID
2. *limit* assertions
    - Every crash occurred during year 2019
3. *intra-record* assertions
    - If a participant record has a Vehicle ID, it should also have a Vehicle Coded Seq#.
    - If a Participant Event Code signifies a severe incident, there must also be an emergency response flag or similar indication.
4. *inter-record check* assertions.
    - Every vehicle listed in the crash data was part of a known crash identified by a Crash ID.
    - No participant should be listed in more than one crash on the same day, assuming data spans short enough periods to avoid double counting.
5. *summary* assertions.
    - The number of crashes recorded is greater than 1,000 but less than 1,000,000.
    - The number of crash records with drug or alcohol involvement reported should not exceed 10% of the total crashes, assuming typical traffic incident statistics.
6. *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”

- The distribution of crashes by time of day should show peaks during typical rush hours.
- The number of crashes per day varies minimally, suggesting an even spread throughout the days of each month.

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.

The code for the assertion is there in the Google colab notebook uploaded in github

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

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

- If a participant record has a Vehicle ID, it should also have a Vehicle Coded Seq#.

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.

The code for the Violated assertion is there in the Google colab notebook uploaded in github

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.

Assertions:

- Every record with alcohol or drug report flags must also have a corresponding BAC Test Results Code if alcohol use is reported.
- Serial numbers must be unique across all crash records to ensure there are no duplicate entries under different Crash IDs.