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 1: I worked on a project called Invoice Payment date prediction system using ML, for this I have dealt with a huge data, I don't say those are errors, but there are empty rows, columns, and null values to overcome this I have done some data preprocessing and imposed missing values to it.

Response 2: I built a data scraper that took data from multiple e-commerce websites. There were certain data that would not match certain fields when scrapped because of the positioning, I had to investigate that and fix them.

Response 3: Yes, previously I have worked on DBMS where we must use a dataset of books that can be used for testing library management systems. In this record, each book is identified by a unique identification number, book title, and author. Each book is assigned its own genre. There is also information about which books are borrowed from which student or teacher. In the raw dataset I found out that there are a lot of columns where there were null values along with redundant data. I separated them into two different tables to make the database simple and more readable. Also, I have an attribute named lost date which will be having null values as it will be only filled in case the book is lost or not returned to the library. So, by analyzing my database I have used the normalization techniques to get refined data.

Response 4: Yes, I've worked with data which had errors. When I took 'Intro to Database Management', I worked on a soccer management system data which I got from an online free resource. Those datasets had multiple redundant data, special characters, blank shells. I've overcome these errors using python, had done some data preprocessing and then used it.

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.

Existence Assertion:

- Each record of type 1 has a serial #
- Every Crash ID has a Crash Date

Limit Assertion:

- All crash day numbers must be a valid day of month.
- Every crash occurred during the year 2019.

Intra Record Assertion:

- If a crash record has a latitude in seconds, it should also have a corresponding longitude in seconds.
- If a crash record has latitude coordinate, then it should also have longitude coordinate.

Inter Record Assertion:

- Vehicles which are involved in the crash but not with an object at the time of the crash.
- Every vehicle id has a participant id.
- Every vehicle listed in the crash data was a part of a known crash.

Summary Assertion:

- Crash Id is unique for all records.
- Most crashes commute between 2PM to 11PM.
- Most crashes involved are commercial vehicles rather than passenger vehicles.
- There are thousands of crashes not millions.

Statistical Assertion:

- Crashes are more likely to occur on weekdays than weekends.
- Crash frequency in winter months is higher than summer months.
- Crashes are uniformly distributed throughout the months of the year.

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 panda Data frame.
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:

- There is a violation for Inter Record assertion which is, "Every vehicle id has a participant id", for this assertion, there is no participant id for the corresponding vehicle id "3442236". I have cross checked with the main csv file, but no records for the participant id are found.

To resolve this violation, there are two approaches to consider among the given options one is **adding the missing value**, this can be done by taking mean/mode/median of the column, since participant id is unique, I don't think this will be a good option to choose. We can also **discard the entire row** since it is a large dataset. Removing the row doesn't affect anything and the assertion will also get passed.

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.