DataEng S23: 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.

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.

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.

Response 1: Me : I worked with 2 data sets earlier. I wouldn't say it had errors, but it needed a little cleaning and preprocessing.It had few empty columns/rows , duplicates and some null values. As part of my project requirements I cleaned them using sql commands.

Response 2: 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 3: Yes I've worked with data which had errors. When I took 'Intro to Databas Management', I worked on a soccer management system data which I got online free resource. That datasets had multiple redundant data, special character , blank shells. I've overcome these errors using python, had done some data preprocessing and then used it

Response 4: As a part of my academic curriculum, I undertook a project that entailed working with a dataset for constructing a database to preserve information concerning movies. During this undertaking, I encountered CSV files that were present with unrelated tables, contained duplicated data, and vacant fields. To resolve these issues, I employed the pandas library, which provided a programming interface for data manipulation and analysis in Python. In addition, I

utilized the numerous data transformation and restructuring features available in Excel, which allowed me to reorganize the rows and columns in a manner that was pertinent to my objectives.

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.

A. 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”
    a. Every crash has a unique identifier(crash ID)
    b. Each crash should have a vehicle ID
2. *limit* assertions. Example: “Every crash occurred during year 2019”
    a. Every crash occurred in HWY 26
3. *intra-record* assertions. Example: “If a crash record has a latitude coordinate then it should also have a longitude coordinate”
    a. Every crash has a crash date
    b. Every person involved in a crash will have an ID
4. Create 2+ *inter-record check* assertions. Example: “Every vehicle listed in the crash data was part of a known crash”
    a. Each crash has at least one vehicle involved
    b. Each participant listed in the crash data was a part of known crash

5. Create 2+ *summary* assertions. Example: “There were thousands of crashes but not millions”
    a. The majority of crashes in the data involve vehicles used to transport students.
    b. Every crash happened in Highway 26
6. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”
    a. Crashes frequently happen during afternoon (middle of the day)
    b. Crashes frequently happened in December due to road and weather conditions

These are just examples. You may use these examples, but you should also create new ones of your own.

B. 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.

C. Run Your Code and Analyze the Results

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

- “Crashes are not evenly/uniformly distributed throughout the months of the year”
- I calculated the number of crashes every month and set a threshold value to 10. And if the difference between max count and min count of the occurrence is greater than the threshold then the assertion fails.

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.

D. 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?

I assumed the general trivia of how and when accidents usually happen as in

- Speeding
- Weather conditions(mostly snow,rain)
- Road conditions
- Reckless Driving
- Light conditions (Night time)

Although some of them turned out to be true, few assumptions were wrong like the light conditions, weather conditions, and speeding.

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

E. 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.

The failed assertion statement that I encountered was taken as part of the example provided in the document. It can be only resolved through change in data provided but not the code..

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.