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 : I have worked with a dataset where we used to face issues with given file containing unreadable characters which would fail our program to execute causing us to manually patch the file remove it and rerun it to resolve the issues and sometimes we faced issue with a numeric currency values where the length exceeded the column allowed prompting us to check for validity of data and ending up correcting the value or if valid changing our db to increase the length for all connected columns and code to pass the files.
Kirk: The dataset I used has a lot of empty values sometimes which required me to discard those data in calculating mean.
Alex: I used a dataset in Data Engineering class and found the breadcrumb data creating a lot of null issues while transmitting to my receiver requiring me to handle the data correctly.
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”
2. *limit* assertions. Example: “Every crash occurred during year 2019”
3. *intra-record* assertions. Example: “If a crash record has a latitude coordinate then it should also have a longitude coordinate”
4. Create 2+ *inter-record check* assertions. Example: “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”
6. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”

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

Answer:

Existence Assertions:

The code contains functions to check that specific columns ('Serial #', 'Crash Month', 'Crash Day', 'Crash Year') are not null for records with a Record Type of 1. This ensures that these essential fields are present for all relevant crash records.

Limit Assertions:

There are functions to validate that the 'Crash Year' is 2019 for records with a Record Type of 1, making sure that all analyzed crash data occurred within the specified year.

Another check ensures that the latitude and longitude degrees are within the typical geographic boundaries for Oregon (latitude between 42 and 46.5, longitude between -124.5 and -116.5) for records with a Record Type of 1. This confirms that the crash locations are within the state's limits.

Intra-Record Assertions:

The code checks whether both specified fields ('City Section ID' and 'County Code') are filled in, implying that if one is present, the other should be as well. This validation ensures that city and county information is consistently recorded.

Inter-Record Check Assertions:

The code checks for Participant IDs for record Type 3 to be unique and then checks for corresponding crash record and checks if the Check that each Crash ID in participant_data matches a Crash ID in crash_data

For each Crash ID, there will be at least two different types of records associated with it.

Summary Assertions:

Find the Total Number of crashes where Fatal Injuries Severity codes Recorded in the crash Data

This assertion checks for the uniqueness of Crash IDs within the subset of crash records categorized as Record Type 1. Initially, the total count of these records is outputted. The assertion then iterates through each record, verifying that each Crash ID appears only once. If a duplicate Crash ID is detected, the code flags the specific instance with its index and Crash ID, and the duplicate record is removed from the dataset. After this validation process, the remaining count of unique Crash IDs is reported, confirming the elimination of any duplicates within the dataset. This ensures that each crash event is uniquely represented in the analysis, an essential factor for accurate crash data reporting and analysis.

Statistical Assertions:

This statistical assertion visually examines the monthly distribution of crash events where Alcohol is involved. A bar chart displays the total number of crashes for each month, highlighted in red, to reveal potential trends or patterns. By analyzing the bar heights, one can infer whether certain months exhibit higher or lower crash frequencies, which could suggest seasonal effects, varying driving conditions, or other time-related factors that influence crash rates.

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:

- Statistical Assertion for Alcohol data using Particpant data failed with 0%
-
-
-

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.

Statistical Assertion for Alcohol data using Particpant data failed because Data did not Exist. I changed the code to use the assertion for alcohol distribution in crash data instead

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.

Statistical Assertion for Alcohol data using Particpant data failed because Data did not Exist. I changed the code to use the assertion for alcohol distribution in crash data instead

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.