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: Yes, I've encountered situations where data had errors. I once worked with a dataset containing customer transaction records. While working in a team, discovered errors when analyzing trends and found inconsistencies in transaction amounts for certain customers. We traced the issue back to data entry errors where decimal points were misplaced or omitted. To rectify this, we implemented data validation checks during the data ingestion process and conducted data cleaning to correct the erroneous entries.

# Background

The data set for this week is [a listing of all Oregon automobile crashes on the Mt. Hood Hwy (Highway 26) during 2019](https://drive.google.com/file/d/1A_R4rDgJsII7wL-onaPeodvv07rPk1SX). This data is provided by the [Oregon Department of Transportation](https://www.oregon.gov/odot) and is part of a [larger data set](https://tvc.odot.state.or.us/tvc/) that is often utilized for studies of roads, traffic and safety.

Here is the available documentation for this data: [description of columns](https://docs.google.com/spreadsheets/d/1G5MV073INT8wSFCzP-sWcPwfiATJpbpj), [Oregon Crash Data Coding Manual](https://www.oregon.gov/ODOT/Data/documents/CDS_Code_Manual.pdf)

Data validation is usually an iterative multi-step process.

1. Create assertions about the data
2. Write code to evaluate your assertions.
3. Run the code, analyze the results
4. 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”

Every crash record includes a data field indicating the Crash Severity.

1. *limit* assertions. Example: “Every crash occurred during year 2019”

The Posted Speed limits are between 0 and 1

1. *intra-record* assertions. Example: “If a crash record has a latitude coordinate then it should also have a longitude coordinate”

Every crash record with a Highway Number should have a corresponding Highway Component specified.

1. Create 2+ *inter-record check* assertions. Example: “Every vehicle listed in the crash data was part of a known crash”

The total count of crashes matches the sum of crash records across all participants involved in those crashes.

Each Vehicle ID listed in the crash data corresponds to at least one Participant ID in the same crash.

1. Create 2+ *summary* assertions. Example: “There were thousands of crashes but not millions”

Every crash occurred on a Highway Number 26

The total number of crashes that happened in 2019 was in the thousands not in millions or billions.

The number of crashes involving fatalities is a smaller percentage compared to the total number of crashes.

1. Create 2+ *statistical distribution assertions*. Example: “crashes are evenly/uniformly distributed throughout the months of the year.”

Crashes are evenly distributed throughout the days of the week, with no significant bias towards any particular day.

The distribution of crashes across months of the year follows a relatively uniform pattern, without significant spikes in any particular month.

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

Code In collab file **‘Data Validation.ipynb’**

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

* #Limit-Assertion:

Assertion Error: Speed Limit values outside expected range (0, 1)

How I resolved: Introduced a null check as some of the records has empty values

* #Inter-record:

AssertionError: Not every Vehicle ID corresponds to at least one Participant ID

How I resolved: Revised assumption

* # Statistical Distribution Assertions Validation

----> 2 assert crash\_data['Week Day Code'].value\_counts().min() > len(crash\_data) / 7 - 100, "Assertion Error: Weekday distribution not uniform"

**3** assert crash\_data['Crash Month'].value\_counts().min() > len(crash\_data) / 12 - 100, "Assertion Error: Monthly distribution not uniform"

AssertionError: Assertion Error: Weekday distribution not uniform

How I resolved: By calculating the standard deviation and asserting the distribution

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.

The violations i resolved in Iteration 1 included using: revise assumptions/assertions, discard the violating row(s), ignore

**Revised and resolved *inter-record check* assertions:**

There are no crashes with negative values for vehicle counts.

Each crash has at least one unique vehicle involved

**Resolved statistical distribution assertion:**

The Weekday Distribution Assertion I asserted in Iteration1 assesses the uniformity of the weekday distribution by examining the variability of crash counts across all weekdays, while the revised assertion focuses on the minimum count of crashes for any weekday compared to an expected minimum count based on uniform distribution assumptions.

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?

Answer:

Through the validation process, I've gained a deeper understanding of the data set's cleanliness and potential issues related to missing values. Specifically, I've learned about:

Null Values: I've observed the presence of null values in the For eg., 'Vehicle ID' column of the vehicles data, indicating potential data entry errors or missing information.

Scaled Values: Some of the columns like age are not easily perceivable. I asserted an age column value range between 16 and 100 as this is the usual range of valid drivers but the column value found to be scaled between 2 and 9.

Data Filtering: I've discovered the need to filter out records with null values like in the 'Vehicle ID' column before saving the vehicles data to a CSV file, ensuring data integrity and consistency.

Data Quality: The validation process has highlighted the importance of data quality checks and the necessity of handling missing values appropriately to maintain the reliability and accuracy of the data set.

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

Iteration 2: Did a first round of iteration on the whole dataset and validated with the assertions. After which the original dataset is transformed into 3 different files Such as Crash.csv, Vehicles.csv and Participants.csv. Also removed records with null Vehicle ID from vehicles\_data. Relevant column data are split and clustered onto these 3 files which aggregate the original dataset.

Step B:

Existence Assertion:

* Ensure that all crash records have a non-null Crash ID.
* Ensure that all crash records have a valid Vehicle ID.

Limit Assertion:

* Limit assertion to check if all crash years are within a certain range, say between 2010 and 2023.

Intra-record Assertions:

* All the Participants ages are scaled between 2 to 9.

Inter-record Check Assertions:

* Ensure that each Vehicle ID corresponds to at least one participant in the participants data.

Summary Assertions:

* Ensure that the total count of vehicles is less than or equal to the total count of crashes.

Statistical Distribution Assertions:

* Check if the distribution of crashes across weekdays follows a pattern.

Step D: Code In collab file **‘Data Split and Validation.ipynb’**

Step C&E: These assertions are violated after validation

Existence Assertion:

* Ensure that all crash records have a valid Vehicle ID.

Fixed by discarding records with null Vehicle ID from vehicles\_data.csv

Intra-record Assertions:

* All the Participants ages are scaled between 2 to 9.

Fixed by doing a null check in the records.