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.

My response:
During my internship at a financial analytics firm, I worked on a Java application for processing financial datasets with thread synchronization to manage data errors like duplicates and missing values. I discovered that inadequate synchronization led to data corruption when multiple threads accessed the data concurrently. To resolve this, I implemented synchronized blocks to ensure that only one thread could modify the data at a time, maintaining data integrity. Additionally, I integrated data validation checks to address duplicates and missing entries, enhancing the reliability of our data processing system. This experience taught me the importance of robust thread management and error handling in high-stakes financial environments.

Response 2(Aditi Simha):
I have worked with a set of data including errors. My job involved converting legacy standalone system codes into web based codes hosted on the Software as a Service layer, such that it can be accessed remotely from all systems, without having the dependency on one single physical system. While doing so, I interacted with REST APIs day-in day-out, making GET, POST and PUT calls and retrieving data from them. The data retrieval majorly involved cleaning the incoming JSON data which would often include noise and corrupted data formats, which had to be transformed into the correct format, before sending them to the next part of the processing pipeline. This involved a lot of String transformations in particular with the help of Regex.

Response 3(Sumana):
I did work with data that included errors. I worked for one of the retail companies which had data from various countries in various languages. Before we use the data in other languages, we had to run a test to check if there were some unwanted characters and remove them before we use them and correct the phrases or words which we previously had stored for that particular language and correct them with an automated script in sfmc.

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 crash occurred on highway

2. ***limit* assertions**:
   -Recorded time of crashes must be valid times during the day 0 to 23.

3. ***intra-record* assertions**:
   - If the 'Alcohol-Involved Flag' is true, there must also be a corresponding police report in the 'Investigating Agency'.

4. ***inter-record check* assertions**:
   -Every 'Participant ID' in Vehicle Records should have a corresponding 'Crash ID' in Crash Records, confirming that every participant is linked to a valid crash event.

   -All records associated with the same 'Crash ID' should be reported by the same investigating agency unless officially transferred to another jurisdiction

5. ***summary* assertions**:
   - More than 5% of recorded crashes involve alcohol

   -The number of fatal crashes is more than 10% of total crashes

6. ***statistical distribution assertions***:
   -Crashes are more frequent on weekdays compared to weekends.

   -Crashes are lesser during rush hours.

C. [MUST] Validate the Assertions

Validation part is done and it is present in this link:
https://github.com/Krishi2026/dataeng_class/tree/main/04_validate

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

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

- intra-record assertions: “If 'Alcohol-Involved Flag' is true, there must also be a corresponding police report in 'Investigating Agency”. None of the rows for the column “Investigating Agency” has values, it is empty. To resolve this missing values should be added.
- Inter-record assertion: “All records associated with the same 'Crash ID' should be reported by the same investigating agency unless officially transferred to another jurisdiction.”. As seen in the data, different investigating agencies are reported within the same Crash ID, which shows that the data does not maintain referential integrity. To resolve this, it should be ensured that the investigating agency should be the same for all the entries having a particular Crash ID.

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.