Geetha Madhuri Chittuluri

DataEng: Data Validation Activity

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 for this assignment, two or more assertions in each category are enough.

1. Create 2+ *existence* assertions.

   a. Every record should have a Crash ID
   b. Every record should have a Record Type
   c. Every vehicle should have a Vehicle ID
   d. Every participant should have a Participlant ID

2. Create 2+ *limit* assertions. The values of most numeric fields should fall within a valid range. Example: “the date field should be between 1/1/2019 and 12/31/2019 inclusive”

   a. The Latitude Degrees field must be between 41 and 46 inclusive
   b. Crash Month field value must be in list 01-12
   c. Crash Hour field should be between 00 and 24 and 99 as a unknown time

3. Create 2+ *intra-record check* assertions.

   a. crash date = crash month + crash day + crash year
   b. Total count of persons involved = Total pedestrian count + Total pedalcyclist count + Total unknown count + Total occupant count

4. Create 2+ *inter-record check* assertions.

   Refer referential integrity assertions

5. Create 2+ *summary* assertions. Example: “every crash has a unique ID”

   a. Every crash has a unique crash id
   b. Every participant has a unique participant id

6. Create 2+ referential integrity insertions. Example “every crash participant has a Crash ID of a known crash”

   a. For every record of type2, vehicle id should not be null
   b. For every record of type3, both vehicle id and participant id should not be null
   c. For every record of type1, serial # should not be null

7. Create 2+ *statistical distribution assertions*.

   a. Crashes are uniformly distributed throughout the year.

B. Validate the Assertions

1. Now study the data in an editor or browser. If you are anything like me you will be surprised with what you find. The Oregon DOT made a mess with their data!
2. Write python code to read in the test data and parse it into python data structures. You can write your code any way you like, but we suggest that you use pandas’ methods for reading csv files into a pandas Dataframe

```
import pandas as pd

# Reading the csv file into a pandas Dataframe
df = pd.read_csv ('Oregon Hwy 26 Crash Data for 2019.csv')
```

3. Write python code to validate each of the assertions that you created in part A. Again, pandas make it easy to create and execute assertion validation code.

   Added “Assertions_Validation_NormalizedData” python code file to Github

4. If you are like me you’ll find that some of your assertions don’t make sense once you actually understand the structure of the data. So go back and change your assertions if needed to make them sensible.

5. Run your code and note any assertion violations. List the violations here.

   For the following assertion I made –
   a. Every vehicle should have a vehicle id – For this assertion I see there are many NAN values

   b. For most of the assertions I made, I see NAN value records

Geetha Madhuri Chittuluri

C. Evaluate the Violations

For any assertion violations found in part B, describe how you might resolve the violation. Options might include “revise assumptions/assertions”, “discard the violating row(s)”, “ignore”, “add missing values”, “interpolate”, “use defaults”, etc.

No need to write code to resolve the violations at this point, you will do that in step E.

If you chose to “revise assumptions/assertions” for any of the violations, then briefly explain how you would revise your assertions based on what you learned.

I normalized the data into 3 dataframes and revised/revalidated those assertions and retrieved the correct the values

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 ABCD iteration?

After analyzing the whole data, I see the whole table data can normalized into 3 different tables and can do validations on those dataframes for accurate results

Next, iterate through the process again by going back to Step A. Add more assertions in each of the categories before moving to steps B and C again. Go through the full loop twice before moving to step E.

Added more assertions and uploaded the python file to github

E. Resolve the Violations

For each assertion violation found during the two loops of the process, write python code to resolve the assertions. This might include dropping rows, dropping columns, adding default values, modifying values or other operations depending on the nature of the violation.

Note that I realize that this data set is somewhat awkward and that it might be best to “resolve the violations” by restructuring the data into proper tables. However, for this

week, I ask that you keep the data in its current overall structure. Later (next week) we will have a chance to separate vehicle data and participant data properly.

Added “Assertions_Validation_NormalizedData” python file to Github

E. Retest

After modifying the dataset/stream to resolve the assertion violations you should have produced a new set of data. Run this data through your validation code (Step B) to make sure that it validates cleanly.

Submit: In-class Activity Submission Form