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.

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:

Dileep Kumar Boyapati: I have worked on the Big bash league dataset. In that dataset I found many null values and missing values in the columns like dismissal_kind, player_dismissed and extras_type etc. To handle these null values and missing values, I deleted the unwanted rows that are not required. After cleaning the data, the model performance got improved.

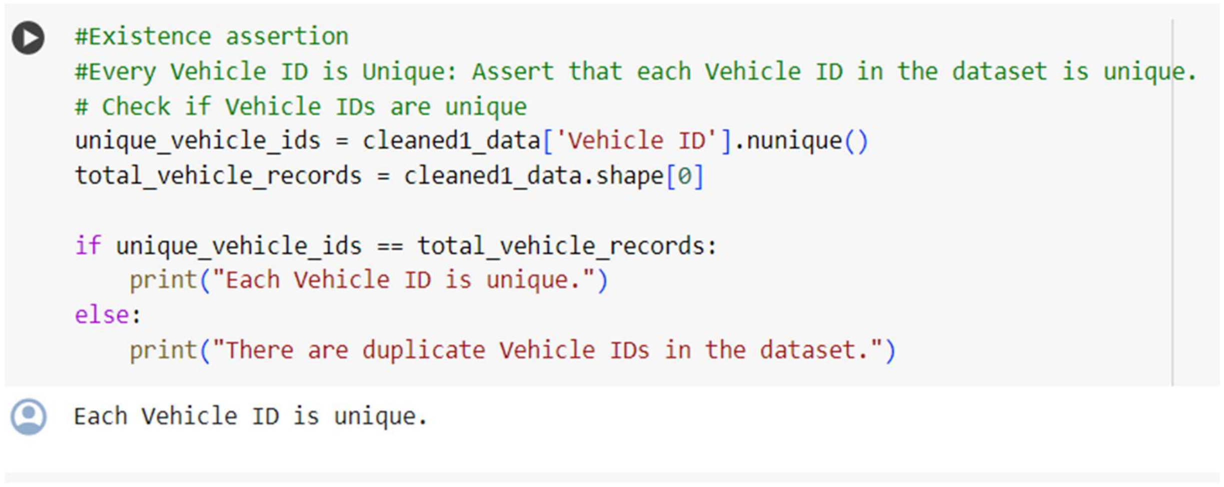
Prathamesh Chakote: I worked on the House price forecast dataset, and there were many nil values in columns like the number of rooms, latitude and longitude, and so on. So, to deal with null values, I eliminated some rows that I believe are unimportant, and I also used Python modules like MICE. While developing a machine learning model, I received less accurate predictions; however, after cleaning and validating the data, the accuracy of my machine learning model improved.

Sai Krishna Mukund: One common issue was dealing with missing values in the dataset. Missing data can affect the performance of machine learning models, so I had to decide on appropriate strategies for handling missing values, such as imputation techniques or removing incomplete records.
As mentioned earlier, the dataset had an imbalance between stroke and non-stroke instances, with a higher proportion of non-stroke cases. Addressing this class imbalance required implementing techniques like SMOTE (Synthetic Minority Over-sampling Technique) to oversample the minority class and ensure balanced training data.

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.

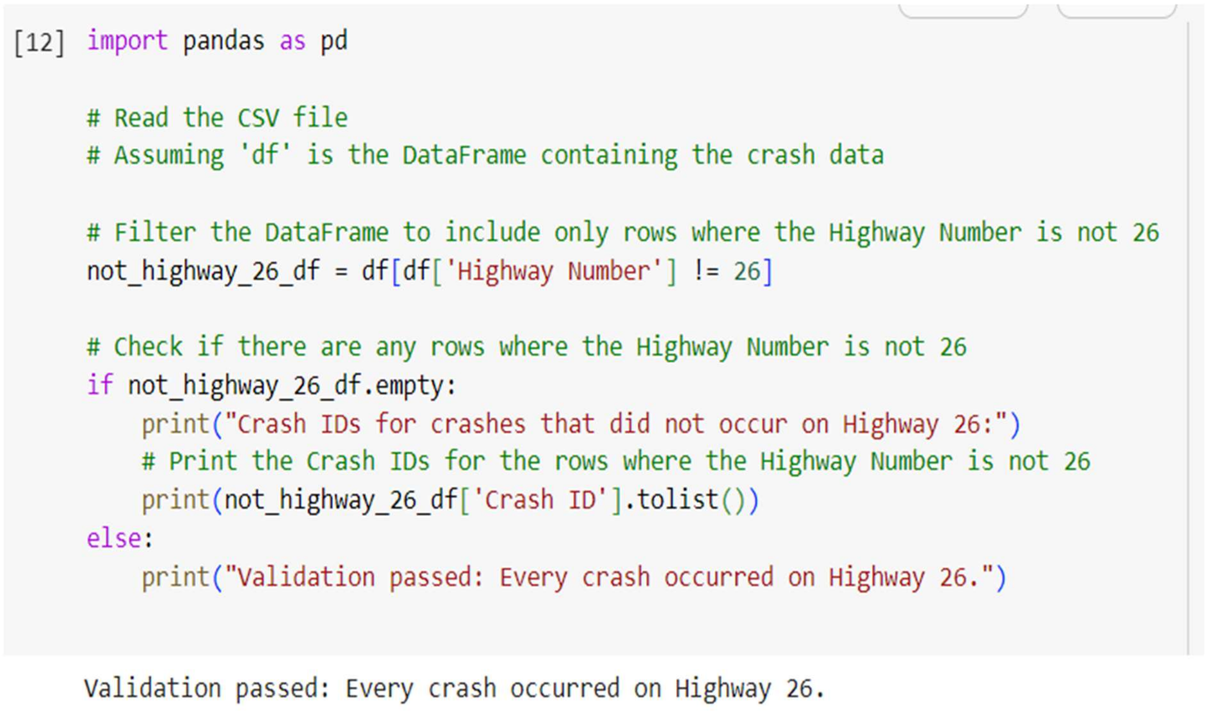
1. *existence* assertions. Example: “Every crash occurred on a date”
    - Every vehicle Id is unique



```
#Existence assertion
#Every Vehicle ID is Unique: Assert that each Vehicle ID in the dataset is unique.
# Check if Vehicle IDs are unique
unique_vehicle_ids = cleaned1_data['Vehicle ID'].nunique()
total_vehicle_records = cleaned1_data.shape[0]

if unique_vehicle_ids == total_vehicle_records:
    print("Each Vehicle ID is unique.")
else:
    print("There are duplicate Vehicle IDs in the dataset.")
```

```
Each Vehicle ID is unique.
```

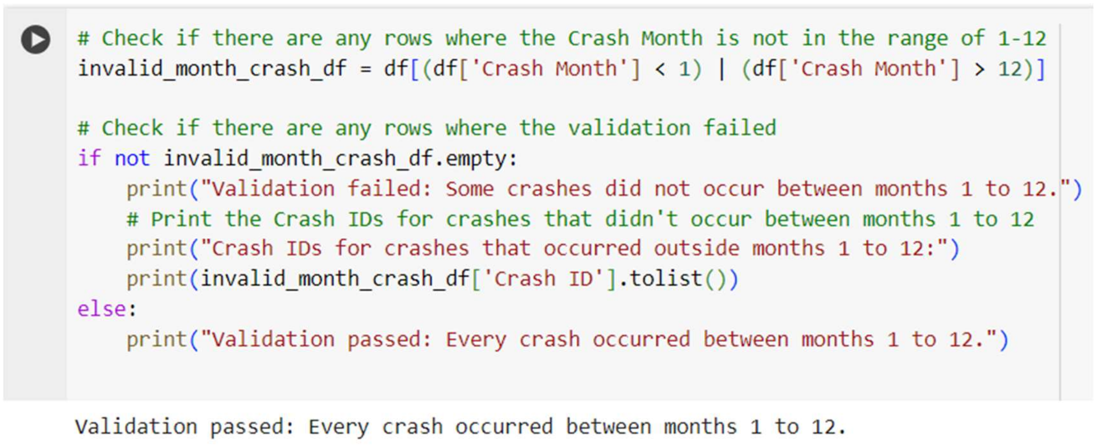
2. *limit* assertions. Example: “Every crash occurred during year 2019”
    - Every crash occurred on Highway 26



```
[12] import pandas as pd

     # Read the CSV file
     # Assuming 'df' is the DataFrame containing the crash data

     # Filter the DataFrame to include only rows where the Highway Number is not 26
     not_highway_26_df = df[df['Highway Number'] != 26]

     # Check if there are any rows where the Highway Number is not 26
     if not_highway_26_df.empty:
         print("Crash IDs for crashes that did not occur on Highway 26:")
         # Print the Crash IDs for the rows where the Highway Number is not 26
         print(not_highway_26_df['Crash ID'].tolist())
     else:
         print("Validation passed: Every crash occurred on Highway 26.")
```

```
Validation passed: Every crash occurred on Highway 26.
```

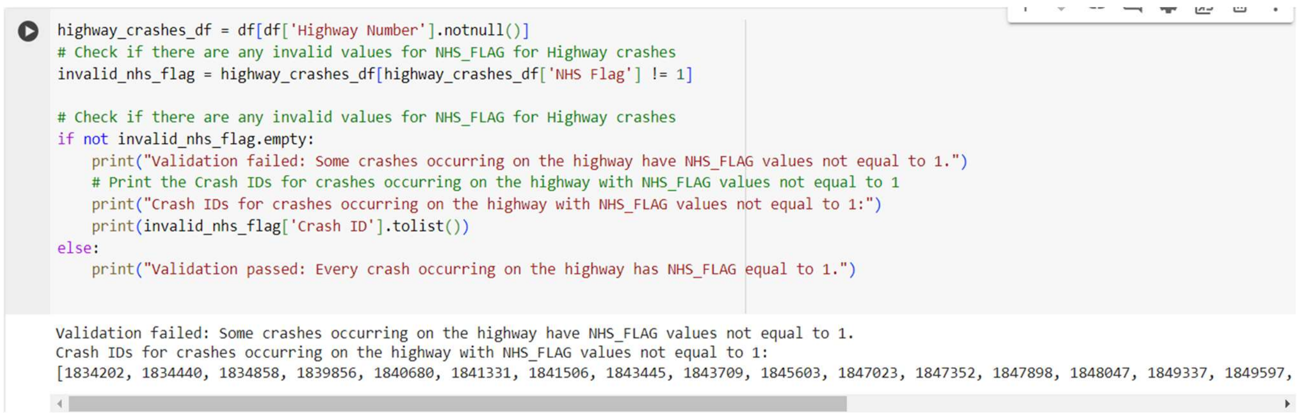
- Every crash should have happened between months 1-12(January-December).



```
# Check if there are any rows where the Crash Month is not in the range of 1-12
invalid_month_crash_df = df[(df['Crash Month'] < 1) | (df['Crash Month'] > 12)]

# Check if there are any rows where the validation failed
if not invalid_month_crash_df.empty:
    print("Validation failed: Some crashes did not occur between months 1 to 12.")
    # Print the Crash IDs for crashes that didn't occur between months 1 to 12
    print("Crash IDs for crashes that occurred outside months 1 to 12:")
    print(invalid_month_crash_df['Crash ID'].tolist())
else:
    print("Validation passed: Every crash occurred between months 1 to 12.")
```

```
Validation passed: Every crash occurred between months 1 to 12.
```

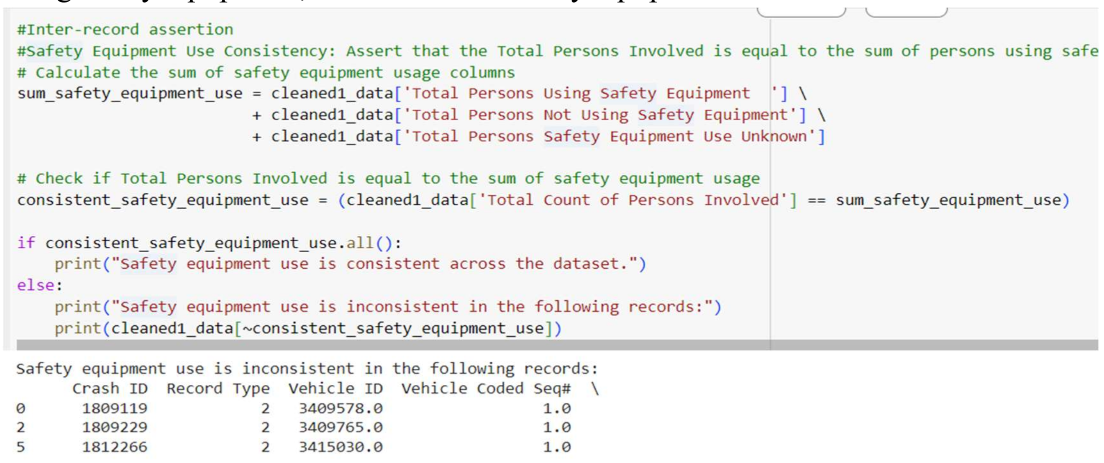
3. *intra-record* assertions. Example: “If a crash record has a latitude coordinate then it should also have a longitude coordinate”
    - For every crash that occurs on the Highway, NHS_FLAG should be 1.



```
highway_crashes_df = df[df['Highway Number'].notnull()]
# Check if there are any invalid values for NHS_FLAG for Highway crashes
invalid_nhs_flag = highway_crashes_df[highway_crashes_df['NHS Flag'] != 1]

# Check if there are any invalid values for NHS_FLAG for Highway crashes
if not invalid_nhs_flag.empty:
    print("Validation failed: Some crashes occurring on the highway have NHS_FLAG values not equal to 1.")
    # Print the Crash IDs for crashes occurring on the highway with NHS_FLAG values not equal to 1
    print("Crash IDs for crashes occurring on the highway with NHS_FLAG values not equal to 1:")
    print(invalid_nhs_flag['Crash ID'].tolist())
else:
    print("Validation passed: Every crash occurring on the highway has NHS_FLAG equal to 1.")
```

```
Validation failed: Some crashes occurring on the highway have NHS_FLAG values not equal to 1.
Crash IDs for crashes occurring on the highway with NHS_FLAG values not equal to 1:
[1834202, 1834440, 1834858, 1839856, 1840680, 1841331, 1841506, 1843445, 1843709, 1845603, 1847023, 1847352, 1847898, 1848047, 1849337, 1849597,
```

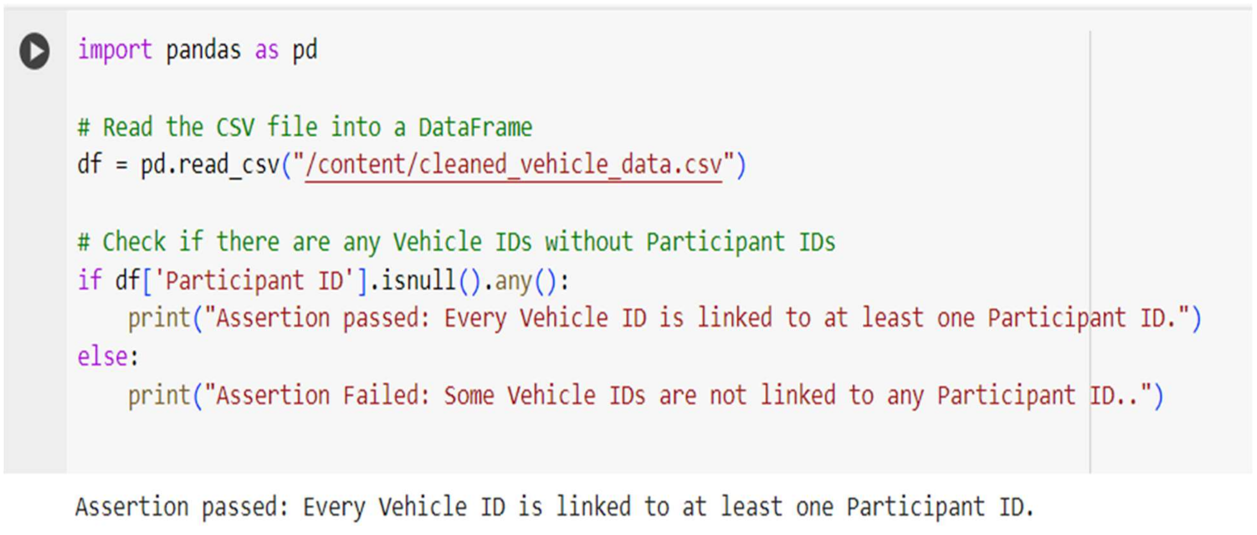
4. Create 2+ *inter-record check* assertions. Example: “Every vehicle listed in the crash data was part of a known crash”
    - Total Persons Involved is equal to the sum of persons using safety equipment, not using safety equipment, and those with safety equipment use unknown.



```
#Inter-record assertion
#Safety Equipment Use Consistency: Assert that the Total Persons Involved is equal to the sum of persons using safe
# Calculate the sum of safety equipment usage columns
sum_safety_equipment_use = cleaned1_data['Total Persons Using Safety Equipment  '] \
                         + cleaned1_data['Total Persons Not Using Safety Equipment'] \
                         + cleaned1_data['Total Persons Safety Equipment Use Unknown']

# Check if Total Persons Involved is equal to the sum of safety equipment usage
consistent_safety_equipment_use = (cleaned1_data['Total Count of Persons Involved'] == sum_safety_equipment_use)

if consistent_safety_equipment_use.all():
    print("Safety equipment use is consistent across the dataset.")
else:
    print("Safety equipment use is inconsistent in the following records:")
    print(cleaned1_data[~consistent_safety_equipment_use])
```

```
Safety equipment use is inconsistent in the following records:
     Crash ID  Record Type  Vehicle ID  Vehicle Coded Seq#  \
0     1809119            2   3409578.0                 1.0
2     1809229            2   3409765.0                 1.0
5     1812266            2   3415030.0                 1.0
```

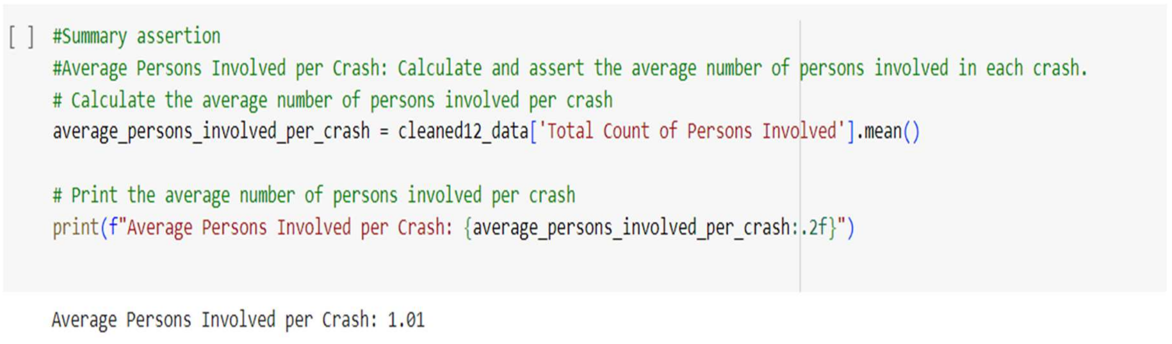
- Every vehicle ID in the crash data is linked to at least one participant ID.



```
import pandas as pd

# Read the CSV file into a DataFrame
df = pd.read_csv("/content/cleaned_vehicle_data.csv")

# Check if there are any Vehicle IDs without Participant IDs
if df['Participant ID'].isnull().any():
    print("Assertion passed: Every Vehicle ID is linked to at least one Participant ID.")
else:
    print("Assertion Failed: Some Vehicle IDs are not linked to any Participant ID..")
```

```
Assertion passed: Every Vehicle ID is linked to at least one Participant ID.
```

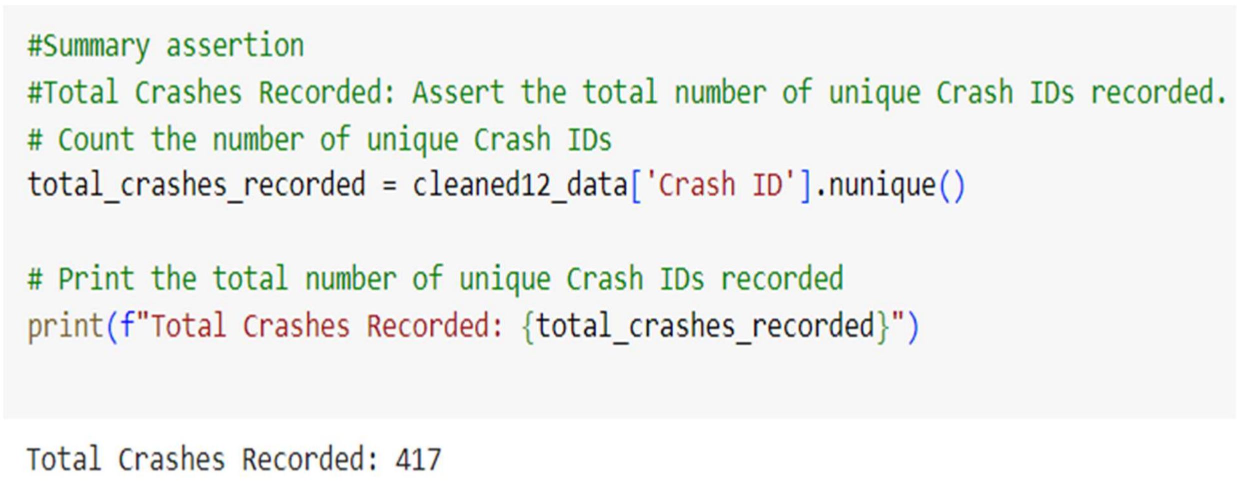
5. Create 2+ *summary* assertions. Example: “There were thousands of crashes but not millions”
    - Verify the average number of persons involved in each crash



```
[ ] #Summary assertion
    #Average Persons Involved per Crash: Calculate and assert the average number of persons involved in each crash.
    # Calculate the average number of persons involved per crash
    average_persons_involved_per_crash = cleaned12_data['Total Count of Persons Involved'].mean()

    # Print the average number of persons involved per crash
    print(f"Average Persons Involved per Crash: {average_persons_involved_per_crash:.2f}")
```

```
Average Persons Involved per Crash: 1.01
```

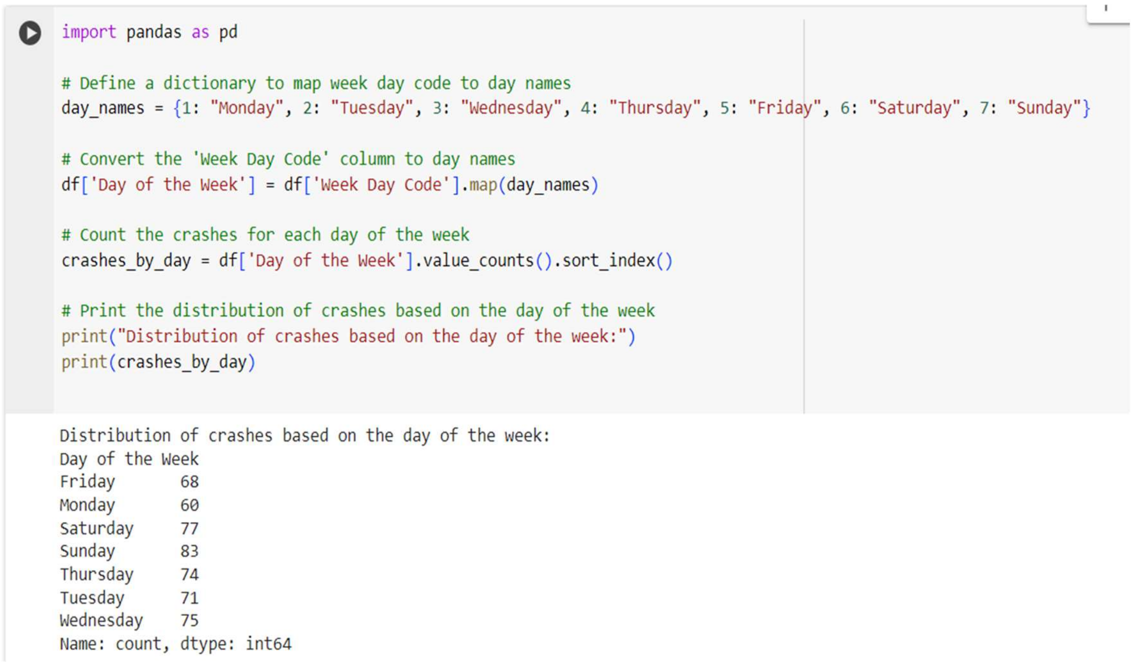
- Number of Unique Crash Ids that are recorded



```
#Summary assertion
#Total Crashes Recorded: Assert the total number of unique Crash IDs recorded.
# Count the number of unique Crash IDs
total_crashes_recorded = cleaned12_data['Crash ID'].nunique()

# Print the total number of unique Crash IDs recorded
print(f"Total Crashes Recorded: {total_crashes_recorded}")
```

```
Total Crashes Recorded: 417
```

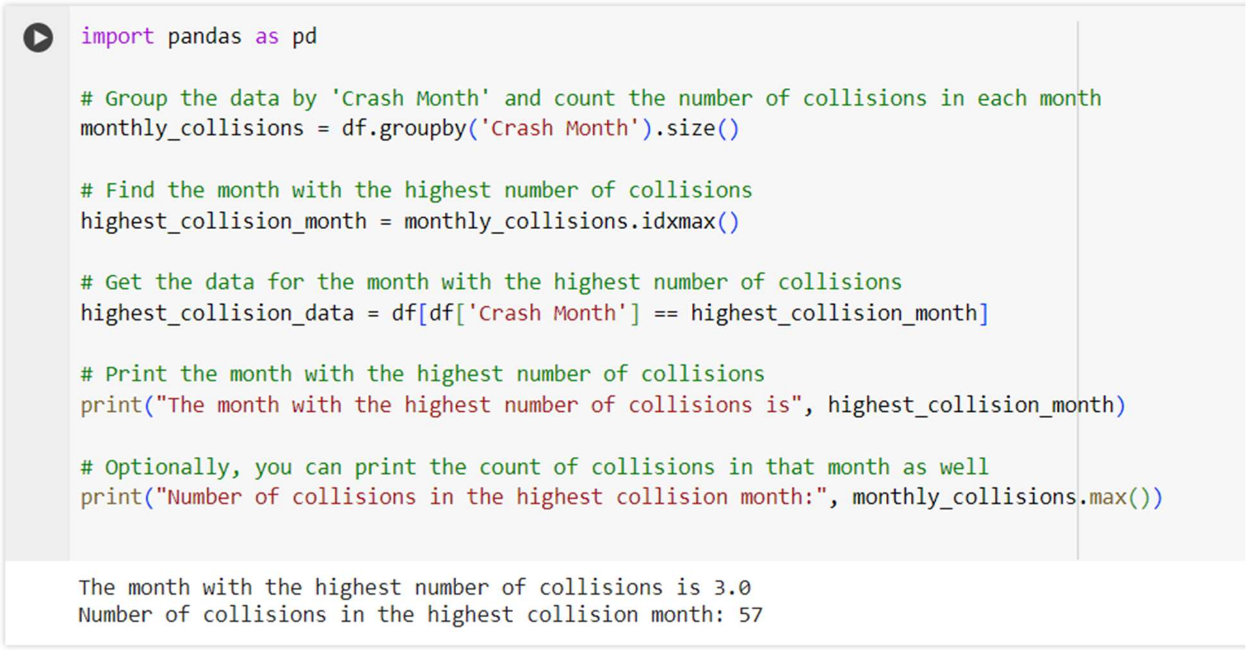
6. Create 2+ *statistical distribution assertions*. Example: "crashes are evenly/uniformly distributed throughout the months of the year."
    - The crash distribution is analyzed based on the day of the week (Monday to Sunday).



```python
import pandas as pd

# Define a dictionary to map week day code to day names
day_names = {1: "Monday", 2: "Tuesday", 3: "Wednesday", 4: "Thursday", 5: "Friday", 6: "Saturday", 7: "Sunday"}

# Convert the 'Week Day Code' column to day names
df['Day of the Week'] = df['Week Day Code'].map(day_names)

# Count the crashes for each day of the week
crashes_by_day = df['Day of the Week'].value_counts().sort_index()

# Print the distribution of crashes based on the day of the week
print("Distribution of crashes based on the day of the week:")
print(crashes_by_day)
```

```
Distribution of crashes based on the day of the week:
Day of the Week
Friday       68
Monday       60
Saturday     77
Sunday       83
Thursday     74
Tuesday      71
Wednesday    75
Name: count, dtype: int64
```

    - Number of Collisions peak in June compared to other months.



```python
import pandas as pd

# Group the data by 'Crash Month' and count the number of collisions in each month
monthly_collisions = df.groupby('Crash Month').size()

# Find the month with the highest number of collisions
highest_collision_month = monthly_collisions.idxmax()

# Get the data for the month with the highest number of collisions
highest_collision_data = df[df['Crash Month'] == highest_collision_month]

# Print the month with the highest number of collisions
print("The month with the highest number of collisions is", highest_collision_month)

# Optionally, you can print the count of collisions in that month as well
print("Number of collisions in the highest collision month:", monthly_collisions.max())
```

```
The month with the highest number of collisions is 3.0
Number of collisions in the highest collision month: 57
```

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.

Answer:

These assertions are validated under B.

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

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

- Some crashes do not have a serial number.
  Solution: Create three different CSV files based on record type, ensuring that each type of data is accurately represented and validated individually.
- Some crashes occurring on the highway have NHS_FLAG values not equal to 1.
  Solution: As our data represents accidents happening on NH 26, discard the rows not following this rule.
- Some entries with 'Speed Involved Flag' equal to 1 have 'State' column values other than 'OR'.
  Solution: Replace 'US' with 'Oregon' as the field that doesn't have 'OR' has 'US' as its value. For null values, add 'OR'.

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.

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.

Answer:
Transformed Everything in part B.

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.

Answer:
After familiarizing myself with the meanings of each column and understanding the significance of the data representation in the form, I partitioned the data into three separate files. This division greatly enhanced my ability to analyze each dataset effectively. Moving forward, I intend to iterate through the entire process again, revisiting steps B, C, D, and E at least once more to ensure a comprehensive understanding and thorough analysis.