SOFE 3290U: Software Quality & Project Management

Lab 5

CRN 75766

Submission for Rivka Sagi

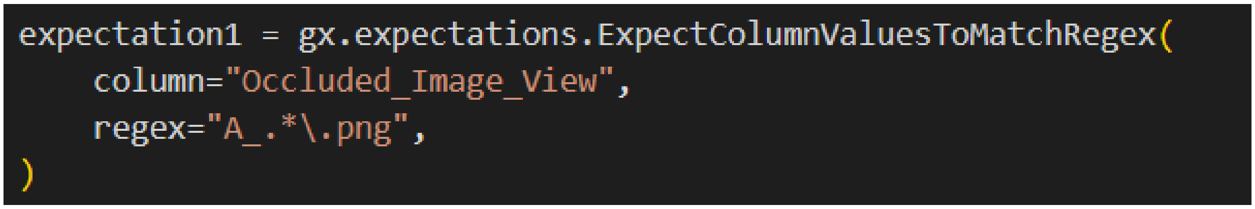
Student #100780926

Github repository:

https://github.com/RivkaRSagi/SOFE3980U_Lab5

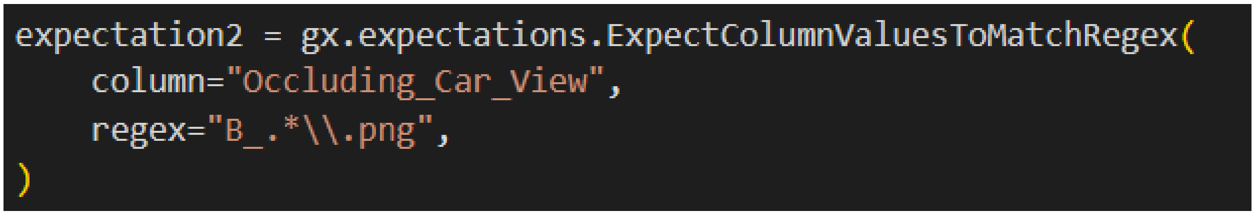
Task 1:

- Expectation 1: Make sure that the Occluded Image View column contains a .png file that begins with A_



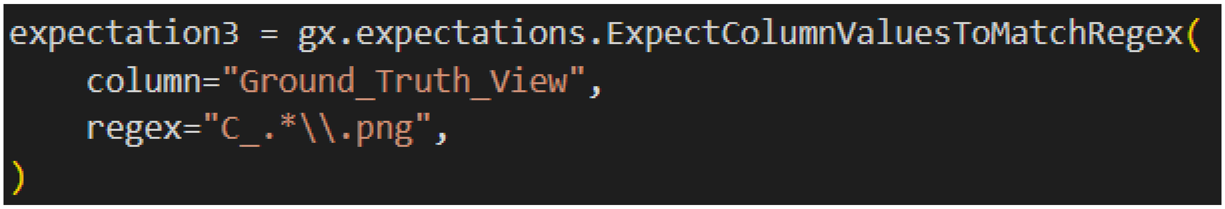
```
expectation1 = gx.expectations.ExpectColumnValuesToMatchRegex(
    column="Occluded_Image_View",
    regex="A_.*\.png",
)
```

- Expectation 2: Make sure that the Occluding Car View column contains a .png file that begins with B_



```
expectation2 = gx.expectations.ExpectColumnValuesToMatchRegex(
    column="Occluding_Car_View",
    regex="B_.*\\.png",
)
```

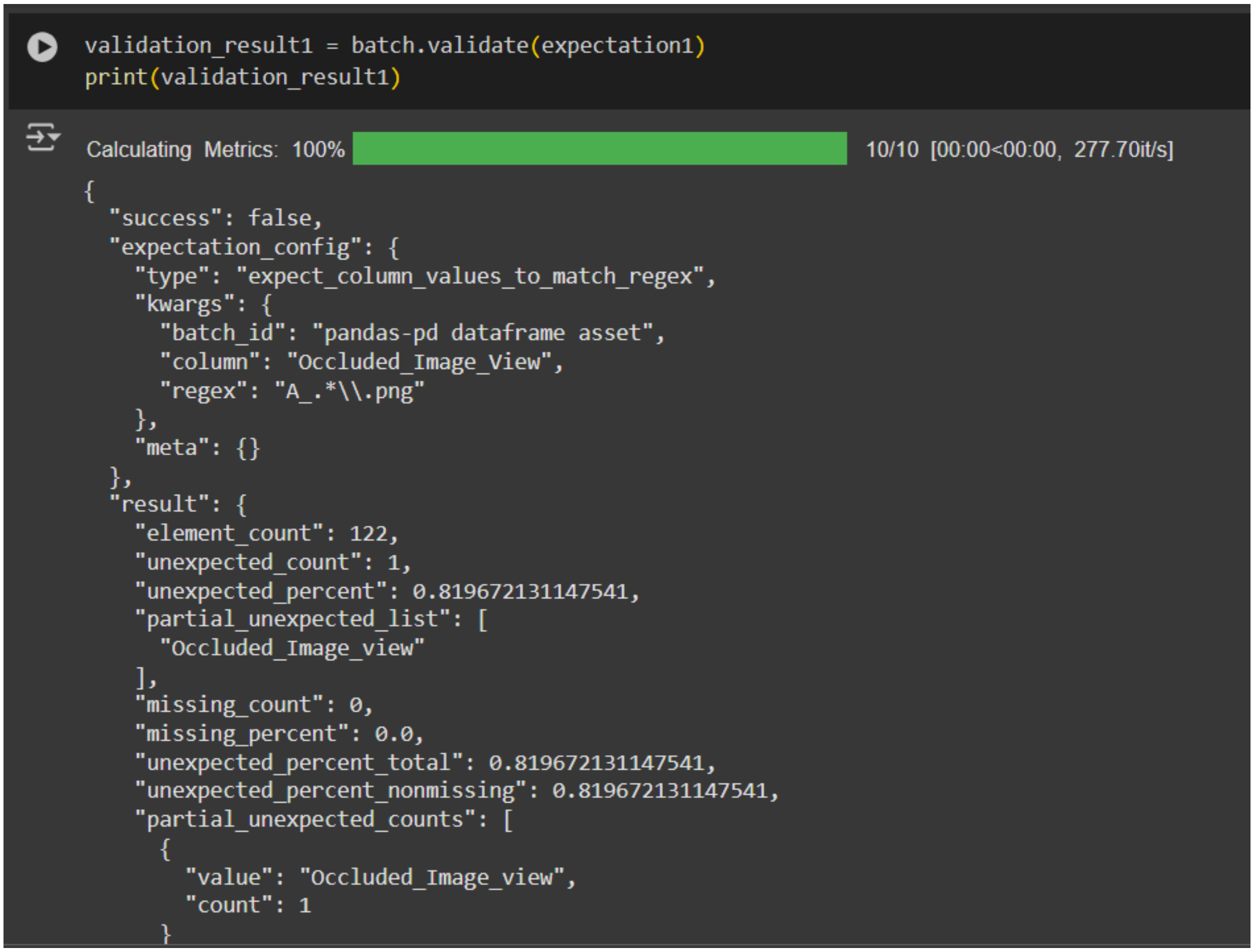
- Expectation 3: Make sure that the Ground Truth View column contains a .png file that begins with C_



```
expectation3 = gx.expectations.ExpectColumnValuesToMatchRegex(
    column="Ground_Truth_View",
    regex="C_.*\\.png",
)
```

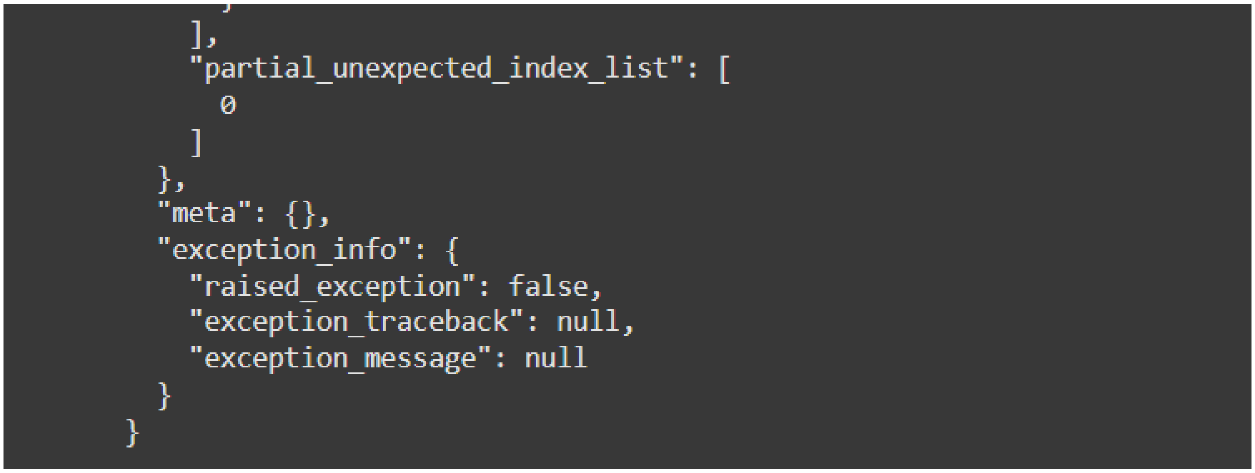
OUTPUT:

- Validation expectation 1:



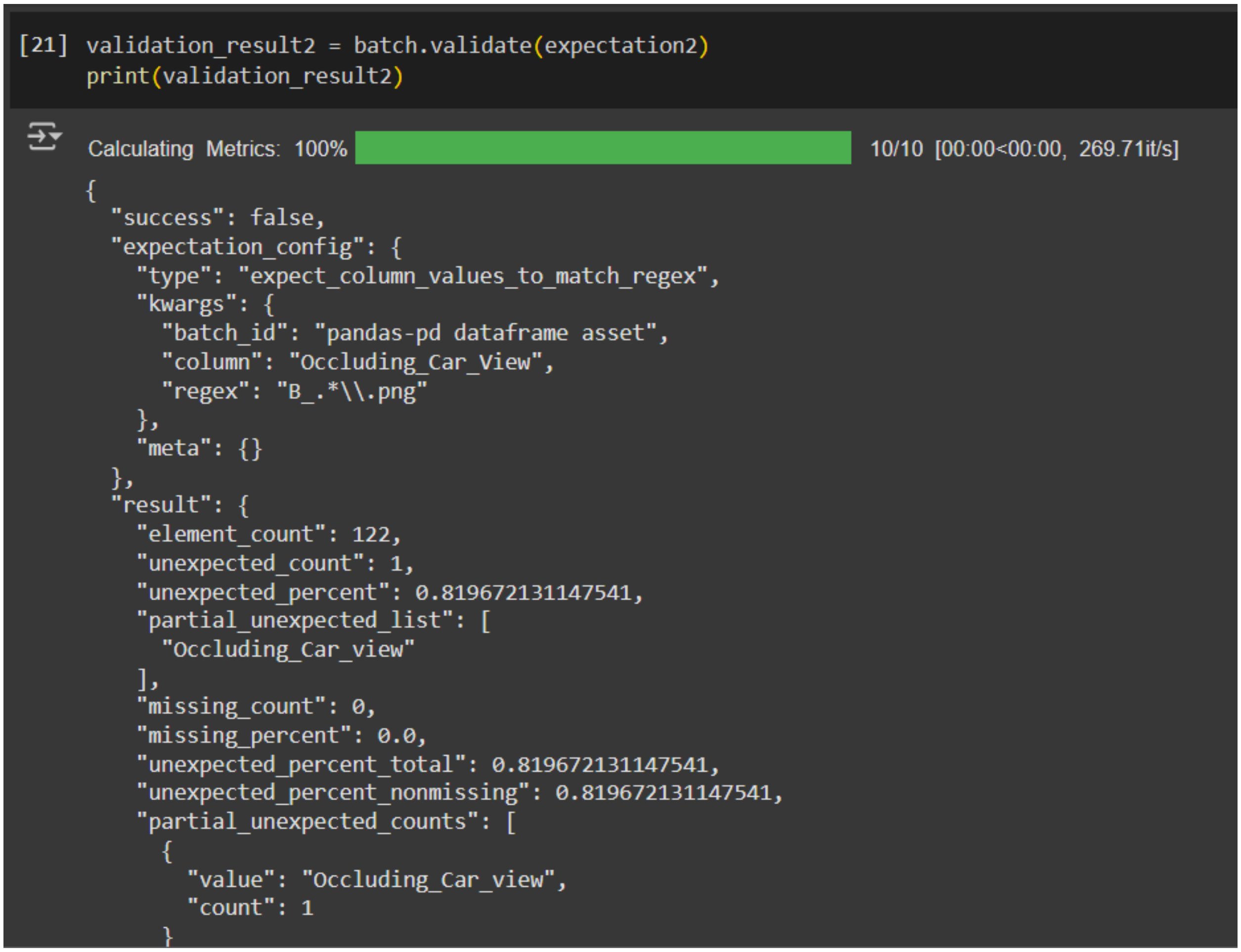
```
validation_result1 = batch.validate(expectation1)
print(validation_result1)
```

```
Calculating Metrics: 100%  10/10 [00:00<00:00, 277.70it/s]
{
  "success": false,
  "expectation_config": {
    "type": "expect_column_values_to_match_regex",
    "kwargs": {
      "batch_id": "pandas-pd dataframe asset",
      "column": "Occluded_Image_View",
      "regex": "A_.*\\.png"
    },
    "meta": {}
  },
  "result": {
    "element_count": 122,
    "unexpected_count": 1,
    "unexpected_percent": 0.819672131147541,
    "partial_unexpected_list": [
      "Occluded_Image_view"
    ],
    "missing_count": 0,
    "missing_percent": 0.0,
    "unexpected_percent_total": 0.819672131147541,
    "unexpected_percent_nonmissing": 0.819672131147541,
    "partial_unexpected_counts": [
      {
        "value": "Occluded_Image_view",
        "count": 1
      }
```



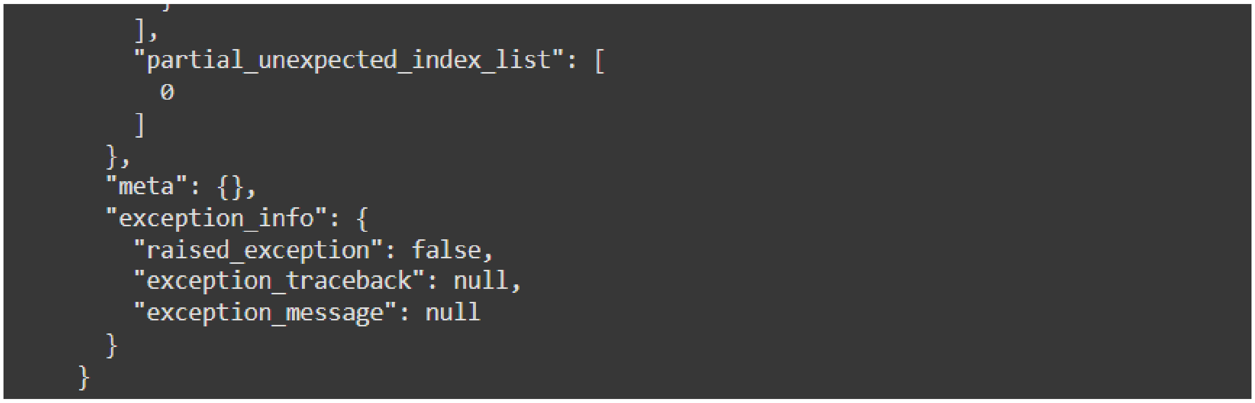
```
    ],
    "partial_unexpected_index_list": [
      0
    ]
  },
  "meta": {},
  "exception_info": {
    "raised_exception": false,
    "exception_traceback": null,
    "exception_message": null
  }
}
```

- Validation expectation 2:



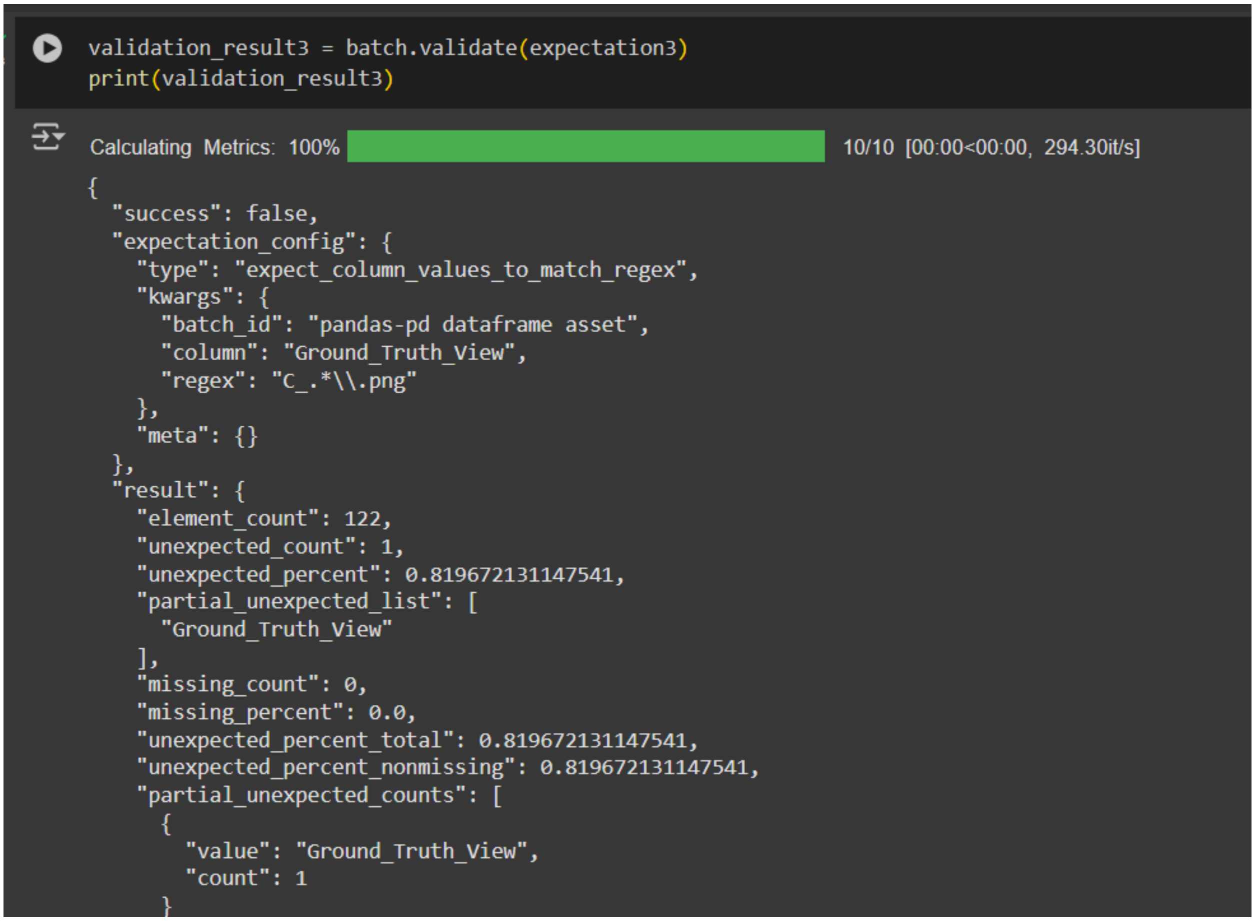
```
[21] validation_result2 = batch.validate(expectation2)
     print(validation_result2)
```

```
Calculating Metrics: 100%   10/10 [00:00<00:00, 269.71it/s]
{
  "success": false,
  "expectation_config": {
    "type": "expect_column_values_to_match_regex",
    "kwargs": {
      "batch_id": "pandas-pd dataframe asset",
      "column": "Occluding_Car_View",
      "regex": "B_.*\\.png"
    },
    "meta": {}
  },
  "result": {
    "element_count": 122,
    "unexpected_count": 1,
    "unexpected_percent": 0.819672131147541,
    "partial_unexpected_list": [
      "Occluding_Car_view"
    ],
    "missing_count": 0,
    "missing_percent": 0.0,
    "unexpected_percent_total": 0.819672131147541,
    "unexpected_percent_nonmissing": 0.819672131147541,
    "partial_unexpected_counts": [
      {
        "value": "Occluding_Car_view",
        "count": 1
      }
```



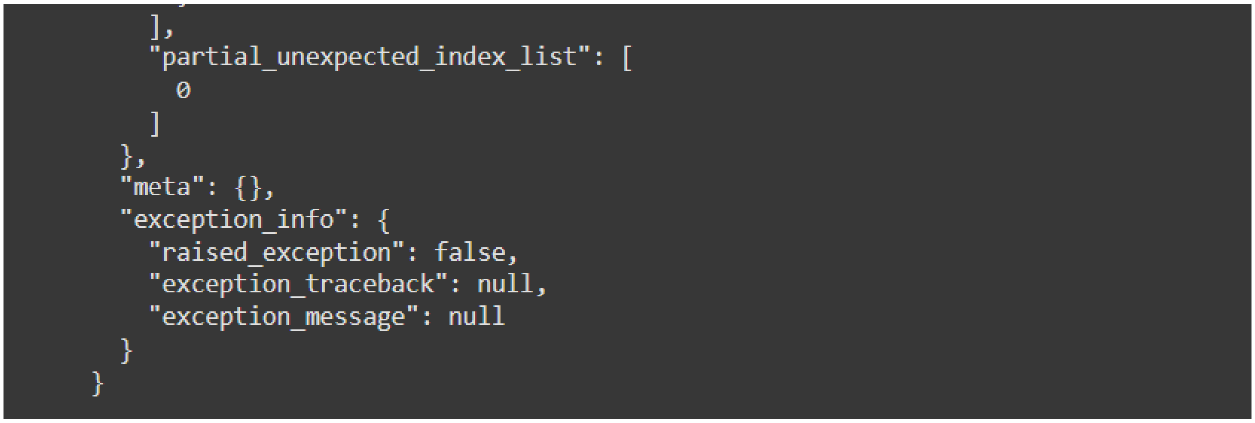
```
    ],
    "partial_unexpected_index_list": [
      0
    ]
  },
  "meta": {},
  "exception_info": {
    "raised_exception": false,
    "exception_traceback": null,
    "exception_message": null
  }
}
```

- Validation expectation 3:



```
validation_result3 = batch.validate(expectation3)
print(validation_result3)
```

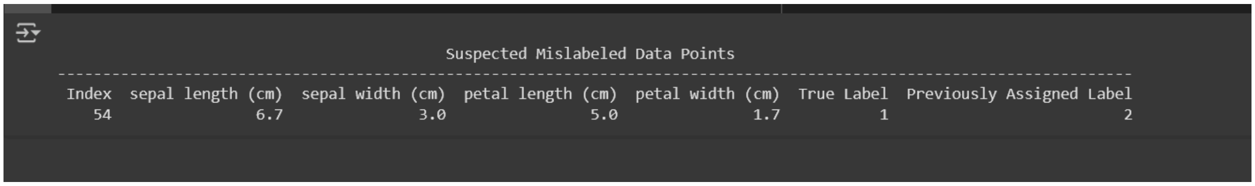
```
Calculating Metrics: 100%  10/10 [00:00<00:00, 294.30it/s]
{
  "success": false,
  "expectation_config": {
    "type": "expect_column_values_to_match_regex",
    "kwargs": {
      "batch_id": "pandas-pd dataframe asset",
      "column": "Ground_Truth_View",
      "regex": "C_.*\\.png"
    },
    "meta": {}
  },
  "result": {
    "element_count": 122,
    "unexpected_count": 1,
    "unexpected_percent": 0.819672131147541,
    "partial_unexpected_list": [
      "Ground_Truth_View"
    ],
    "missing_count": 0,
    "missing_percent": 0.0,
    "unexpected_percent_total": 0.819672131147541,
    "unexpected_percent_nonmissing": 0.819672131147541,
    "partial_unexpected_counts": [
      {
        "value": "Ground_Truth_View",
        "count": 1
      }
```



```
    ],
    "partial_unexpected_index_list": [
      0
    ]
  },
  "meta": {},
  "exception_info": {
    "raised_exception": false,
    "exception_traceback": null,
    "exception_message": null
  }
}
```

Task 2:

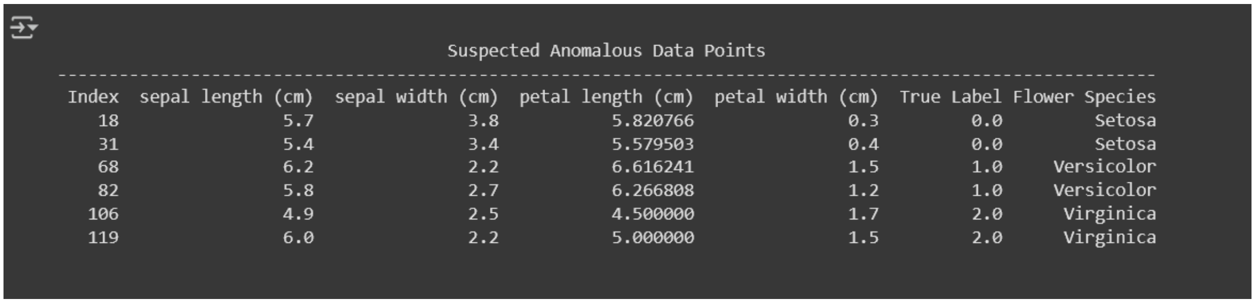
After running all the cells the final output is:



```
                                  Suspected Mislabeled Data Points
 ----------------------------------------------------------------------------------------------------
 Index  sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  True Label  Previously Assigned Label
    54                6.7               3.0                5.0               1.7           1                          2
```

Based on the context of this task, the iris dataset is meant to label a data point as a specific type of iris based on the measurements of the flower recorded for that data point. Earlier in the task some of the data was randomly mislabelled in order to simulate mistakes in the model. The dataset is then run through a classifier wrapped in a cleanLearning cleaner to detect and correct mislabeled data **up to 100 datapoints**. The model is retrained and then checked for mislabeled data points. The detected mislabeled data point must have been from the remaining 50 data points that came after the cleaned up 100 points defined in the classifier.

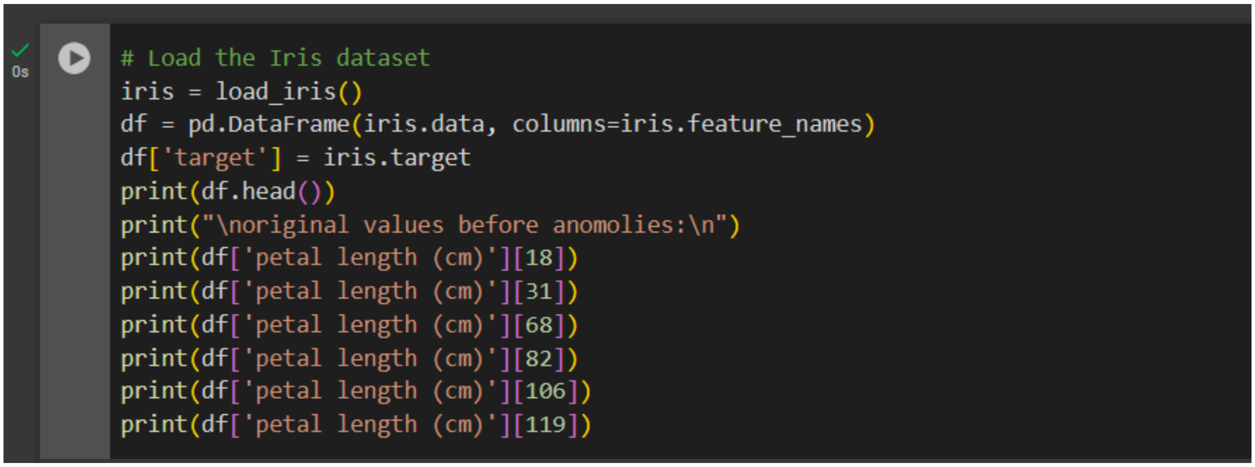
Task 3:



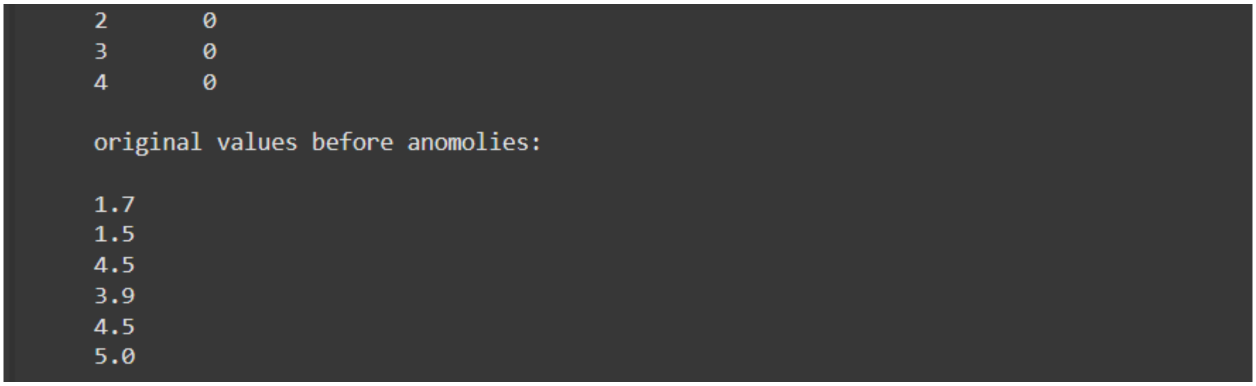
```
                                  Suspected Anomalous Data Points
 ----------------------------------------------------------------------------------------------------
 Index  sepal length (cm)  sepal width (cm)  petal length (cm)  petal width (cm)  True Label Flower Species
    18                5.7               3.8           5.820766               0.3         0.0         Setosa
    31                5.4               3.4           5.579503               0.4         0.0         Setosa
    68                6.2               2.2           6.616241               1.5         1.0     Versicolor
    82                5.8               2.7           6.266808               1.2         1.0     Versicolor
   106                4.9               2.5           4.500000               1.7         2.0      Virginica
   119                6.0               2.2           5.000000               1.5         2.0      Virginica
```

The anomalous data points have manipulated values for the petal length, which should have a significant impact on the label for that data point. The setosa flower should have petal length values roughly below 2cm, the versicolor flower should have a petal length roughly between 2 and 3.5 cm, and the virginica flower should have petal length values higher than 3.5 cm.

To compare the differences between the dataset before and after the dataset, I added print statements for the petal length at the changed points (indices 18, 31, 68, 82, 106, 119) just after loading the iris dataset.



```python
# Load the Iris dataset
iris = load_iris()
df = pd.DataFrame(iris.data, columns=iris.feature_names)
df['target'] = iris.target
print(df.head())
print("\noriginal values before anomolies:\n")
print(df['petal length (cm)'][18])
print(df['petal length (cm)'][31])
print(df['petal length (cm)'][68])
print(df['petal length (cm)'][82])
print(df['petal length (cm)'][106])
print(df['petal length (cm)'][119])
```



```
2       0
3       0
4       0

original values before anomolies:

1.7
1.5
4.5
3.9
4.5
5.0
```

The data points 18 and 31 originally had petal lengths of 1.7 and 1.5, and the anomalous data changed the values to 5.82.. and 5.57.., which is a significant change and would make a difference in the label for those points, which is supposed to be Setosa. Points 68 and 82 originally had lengths 4.5 and 3.9, and were changed to 6.61 and 6.26, which is also a significant difference. The last two points detected as anomalies actually did not change from the original dataset, indicating that they are not actually anomalies but may have been outliers in the original data.