Installation/Execution Guide:

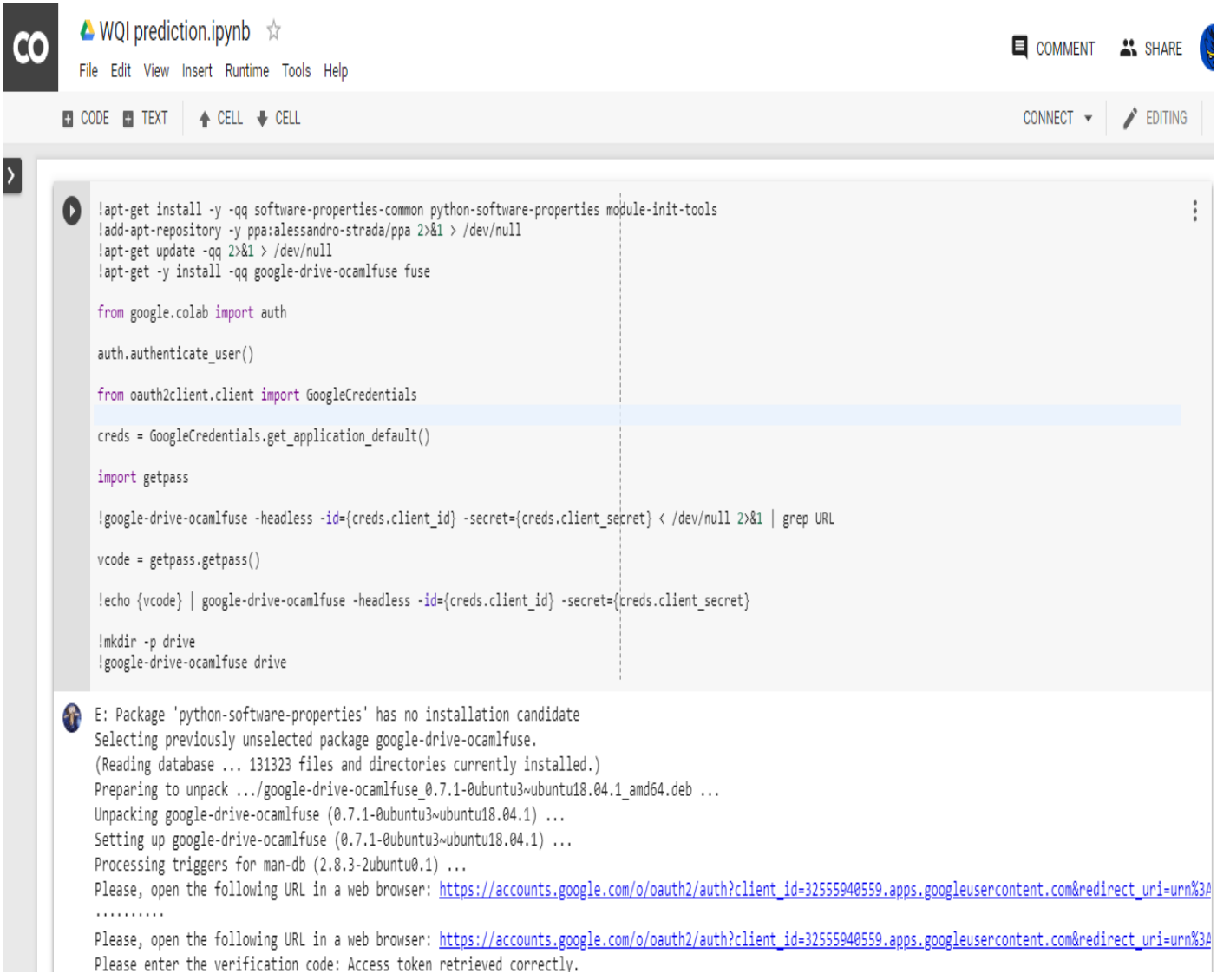
Step1: Importing Data from MongoDB

```
mongoimport --db documents --collection catalog --type csv --
headerline --file /data/dump/water/WaterqualityIndex.csv
```

Step2: Query the collection

```
> db.catalog.find()
```

Step3: Once File is exported to the documents and uploaded into the Drive. Open the file with Google COlab along with Python3 Notebook



Step 4: Change the Directory where the csv file is stored

```
cd /content/drive/WQI\ Estimation
```

```
/content/drive/WQI Estimation
```

Step 5: Import the data to Workbook

```
import pandas as pd

WQI_parameters = pd.read_csv('WaterQualityData.csv')
df = pd.DataFrame(WQI_parameters, columns=["pH",  "TN", "BOD5", "TP", "NH3+", "COD", "Iron",  "Copper", "Zinc", "DO", "TDS",  "Ca", "Mg", "Na", "Cl-",  "HCO", "SO4",
print (df)
```

```
      pH    TN  BOD5    TP  NH3+     COD  Iron  Copper  Zinc     DO  \
0   8.31  0.87  6.24  0.69  5.44   77.31  0.35    1.38  1.74   9.48
1   8.01  6.38  2.90  0.20  0.38   11.00  0.58    0.97  2.68   8.95
2   7.99  5.92  3.18  0.06  8.32   75.80  0.81    0.40  3.18   6.27
3   8.41  0.64  2.88  0.31  3.78  164.37  0.56    0.45  2.50   7.53
4   8.39  0.95  6.50  0.37  0.61  107.69  0.04    0.16  1.03   2.90
5   8.00  4.54  3.57  0.49  8.04  163.94  0.69    1.18  1.94   3.16
6   8.24  1.62  3.71  0.47  6.12  161.91  1.29    0.81  0.27   9.63
7   7.96  1.23  3.52  0.81  2.05    7.06  0.82    0.74  0.18   8.04
8   8.38  0.71  4.47  0.06  1.93   42.34  0.30    0.99  0.12   7.31
9   7.80  5.10  4.73  0.67  2.73   47.47  0.50    0.80  1.34   6.31
10  8.07  4.49  6.84  0.65  1.00   45.88  1.12    0.25  3.01   9.69
11  8.18  6.98  7.75  0.31  2.61   21.10  0.68    0.89  0.75   7.77
12  8.43  3.20  5.39  0.29  2.95  102.53  1.20    1.74  1.37   4.00
13  8.06  4.32  6.75  0.12  6.09  147.05  0.38    1.88  3.28   7.49
14  7.67  5.69  4.97  0.36  3.62   44.79  0.69    1.81  2.87   9.41
15  8.30  2.08  3.58  0.25  0.23  146.90  0.33    0.27  3.17   2.22
16  8.35  2.52  3.72  0.27  7.19  149.37  0.20    1.21  2.08   6.07
17  7.90  4.67  2.41  0.16  3.28   78.68  0.08    0.41  1.31   7.93
```

Step 6: Set a Plot point on Data

```
df = df.fillna(0)
```

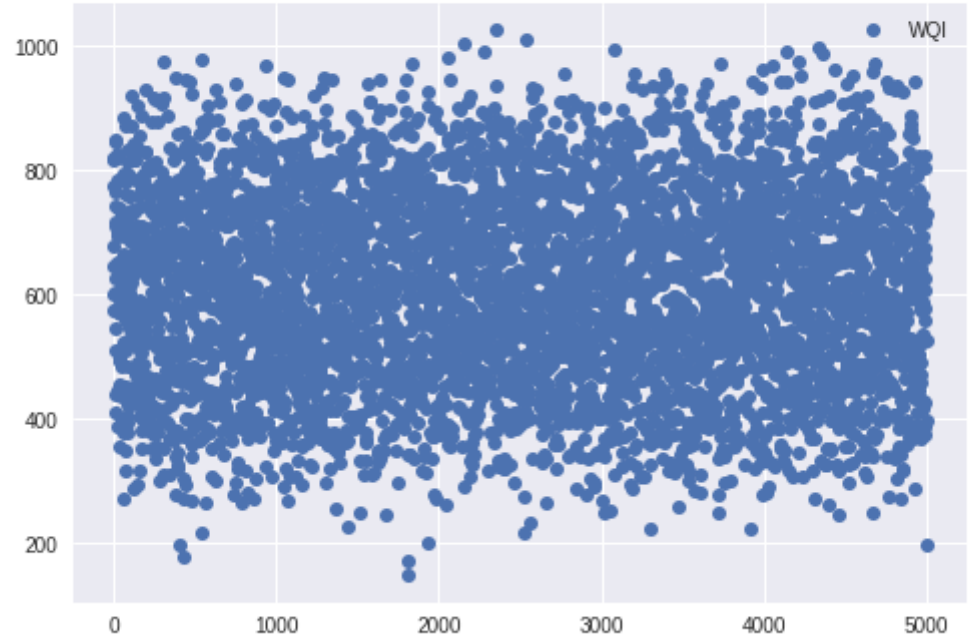
Step 7: Calculate Water Quality Index using the values and Formula

```
import numpy as np

W = np.array([0.072, 0.054, 0.091, 0.054, 0.054, 0.072, 0.018, 0.018, 0.036, 0.018, 0.018, 0.036, 0.036, 0.036, 0.054, 0.054, 0.072, 0.091, 0.018])
S = np.array([7.65, 0.50, 3.00, 0.10, 0.50, 15.00, 0.30, 1.00, 1.00, 6.00, 450.00, 300.00, 30.00, 200.00, 250.00, 1, 250.00, 50.00, 0.05])

df_y = pd.DataFrame(columns=['WQI'])

for index, row in df.iterrows():
  C = np.array(row)
  Q = (C / S) * 100
  SI = W * Q
  df_y.loc[index] = [np.sum(SI)]
```

Step 8: Plot the graph to check the Scatter

```
df_y.plot(style=["o"])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1fff262c50>
```



Step 9: Check the Index Quality

```
[ ]  df_y[(df_y['WQI'] > 800.00)].count()
```

```
WQI    577
dtype: int64
```

Step 10: Import **Keras** to model the Data

```
[ ]  import numpy
     import pandas
     from keras.models import Sequential
     from keras.layers import Dense
     from keras.wrappers.scikit_learn import KerasRegressor
     from sklearn.model_selection import cross_val_score
     from sklearn.model_selection import KFold
     from sklearn.preprocessing import StandardScaler
     from sklearn.pipeline import Pipeline
```

```
[ ]  X = df
     Y = df_y
```

Step 11: Create a Base Model to check the Error in WQI

```
# define base model
def baseline_model():
  # create model
  model = Sequential()
  model.add(Dense(19, input_dim=19, kernel_initializer='normal', activation='relu'))
  model.add(Dense(1, kernel_initializer='normal'))
  # Compile model
  model.compile(loss='mean_squared_error', optimizer='adam')
  return model
```

```
# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
# evaluate model with standardized dataset
estimator = KerasRegressor(build_fn=baseline_model, epochs=100, batch_size=5, verbose=0)
```

```
kfold = KFold(n_splits=10, random_state=seed)
results = cross_val_score(estimator, X, Y, cv=kfold)
print("Results: %.2f (%.2f) MSE" % (results.mean(), results.std()))
```

```
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/framework/op_def_library.py:263: colocate_with (from tensorflo
Instructions for updating:
Colocations handled automatically by placer.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/ops/math_ops.py:3066: to_int32 (from tensorflow.python.ops.mat
Instructions for updating:
Use tf.cast instead.
Results: -7.88 (17.24) MSE
```

Step 12: Evaluate the model by creating Standardised dataset

```
# evaluate model with standardized dataset
numpy.random.seed(seed)
estimators = []
estimators.append(('standardize', StandardScaler()))
estimators.append(('mlp', KerasRegressor(build_fn=baseline_model, epochs=50, batch_size=5, verbose=0)))
pipeline = Pipeline(estimators)
kfold = KFold(n_splits=10, random_state=seed)
results = cross_val_score(pipeline, X, Y, cv=kfold)
print("Standardized: %.2f (%.2f) MSE" % (results.mean(), results.std()))
```

```
Standardized: -0.20 (0.11) MSE
```

Step 13: Apply same steps to create a Larger Model and check the Accuracy of WQI

```
[ ]  def larger_model():
       # create model
       model = Sequential()
       model.add(Dense(19, input_dim=19, kernel_initializer='normal', activation='relu'))
       model.add(Dense(9, kernel_initializer='normal', activation='relu'))
       model.add(Dense(1, kernel_initializer='normal'))
       # Compile model
       model.compile(loss='mean_squared_error', optimizer='adam')
       return model
```

```
[ ]  numpy.random.seed(seed)
     estimators = []
     estimators.append(('standardize', StandardScaler()))
     estimators.append(('mlp', KerasRegressor(build_fn=larger_model, epochs=50, batch_size=5, verbose=0)))
     pipeline = Pipeline(estimators)
     kfold = KFold(n_splits=10, random_state=seed)
     results = cross_val_score(pipeline, X, Y, cv=kfold)
     print("Larger: %.2f (%.2f) MSE" % (results.mean(), results.std()))
```

```
Larger: -2.48 (3.49) MSE
```

Conclusion: Thus, with the help of all the models the WQI is calculated and standard value is being generated and this value can be compared with the WHO Water Quality Standards to check the Purity.