**Code Python Script: DARRAJEC2Q1.py**

**# -\*- coding: utf-8 -\*-**

import pandas

import statistics

from textblob import TextBlob

# global function for reading data from excel file using Pandas Library

def get\_excel\_rows():

try:

file\_name = 'rest.xlsx'

read\_cols = ['stars', 'review\_text']

data\_frame = pandas.read\_excel(file\_name, usecols=read\_cols, dtype={'stars': float})

return data\_frame.to\_dict(orient='record')

except FileNotFoundError as err:

print("Please provide correct file path: {}".format(err))

class DARRAJEC2Q1:

records = get\_excel\_rows()

stars\_dataset = []

polarities\_dataset = []

# class constructor

def \_\_init\_\_(self):

# calculating polarity for each user review\_text and make data sets

for row in self.records:

polarity = self.calculate\_polarity(row['review\_text'])

polarity = round(polarity, 2)

self.polarities\_dataset.append(polarity)

self.stars\_dataset.append(row['stars'])

def calculate\_polarity(self, user\_review\_text):

# for calculating review polarity using TextBlob Statistic Library

return TextBlob(user\_review\_text or '').polarity

def calculate\_standard\_deviation(self, data\_set):

return statistics.stdev(data\_set)

def get\_min\_stars\_rating(self):

# return min(self.stars\_dataset)

# OR

min\_stars = 6

for row in self.records:

# checking data type should be integer or float

if not isinstance(row['stars'], (int, float)):

continue

if row['stars'] < min\_stars:

min\_stars = row['stars']

return min\_stars

def get\_max\_stars\_rating(self):

# return max(self.stars\_dataset)

# OR

max\_stars = 0

for row in self.records:

# checking stars column value data\_type should be integer or float

if not isinstance(row['stars'], (int, float)):

continue

if row['stars'] > max\_stars:

max\_stars = row['stars']

return max\_stars

def get\_stars\_average(self):

# avg = sum(self.stars\_dataset) / len(self.stars\_dataset)

# OR

avg = 0

values\_count = 0

for row in self.records:

# checking data type should be integer or float

if not isinstance(row['stars'], (int, float)):

continue

avg += row['stars']

values\_count += 1

avg = avg / values\_count

return round(avg, 2)

def get\_stars\_range(self):

return f'({self.get\_min\_stars\_rating()} to {self.get\_max\_stars\_rating()})'

def get\_min\_polarity(self):

return min(self.polarities\_dataset)

def get\_max\_polarity(self):

return max(self.polarities\_dataset)

def get\_polarity\_range(self):

return f'({self.get\_min\_polarity()} to {self.get\_max\_polarity()})'

def get\_polarity\_average(self):

return round(sum(self.stars\_dataset) / len(self.stars\_dataset), 2)

def get\_stars\_standard\_deviation(self):

return self.calculate\_standard\_deviation(self.stars\_dataset)

def get\_polarities\_standard\_deviation(self):

return self.calculate\_standard\_deviation(self.polarities\_dataset)

def display\_results(self):

# Stars analysis results

print(f"Minimum Stars = {self.get\_min\_stars\_rating()}")

print(f"Maximum Stars = {self.get\_max\_stars\_rating()}")

print(f"Average Rating = {self.get\_stars\_average()}")

print(f"Stars Range = {self.get\_stars\_range()}")

# Polarities analysis results

print(f"\nMinimum Polarity = {self.get\_max\_polarity()}")

print(f"Maximum Polarity = {self.get\_max\_polarity()}")

print(f"Average Polarity = {self.get\_polarity\_average()}")

print(f"Polarity Range = {self.get\_polarity\_range()}")

# Standard Deviation Results for both Stars and Polarities DataSets

print(f"\nStars Standard Deviation = {self.get\_stars\_standard\_deviation()}")

print(f"Polarities Standard Deviation = {self.get\_polarities\_standard\_deviation()}")

# main

if \_\_name\_\_ == "\_\_main\_\_":

# create class object

darraj\_object = DARRAJEC2Q1()

# call function using class object for displaying results

darraj\_object.display\_results()

# The END

**Output ScreenShot DARRAJEC2Q1.py:-**

