Radial Analytics Assessment

By- Aastha Grover

Note: I have chosen python language to work on the dataset provided.

Instructions for executing the Python Code (On Windows):

1. Clone the directory from Git or download the zip file and extract the contents.
2. Go to command line and change the present working directory to the directory where 'Question1' is placed.
3. In the command line, write 'python Question1.py'. It will prompt you for the path of input file that is 'data.csv' file.
4. There is an 'Input Data' folder in the submission. Please copy the file path till 'Input Data'. Example: `'C:\Users\aasth\Desktop\AllSubjects\Radial Analytics assessmnt'`.

   Change the forward '\' to backward '/'.
   `'C:/Users/aasth/Desktop/AllSubjects/Radial Analytics assessmnt'`

5. Copy and paste the above path in the command line where it prompted you for input path of 'data.csv' file.
6. Press 'Enter' and the 'output1.csv' will be created in the present working directory 'Question2'.
7. For 'Question2' change the present working directory to the directory where 'Question2' is placed.
8. In the command line, write 'python Question2.py'. It will prompt you for the path of input file that is 'data.csv' file.
9. Repeat steps 4 & 5 and the file 'output2.csv' will be created in the present working directory 'Question2'.

Note: Python Code is indented. Any changes in the file can cause issues with execution of the output.

Question 1:

1) Defined a function for reading the CSV file and handled the Exception using try except block.

```
# Question 1

#Function to read data from the csv file
def readFile():
    import os
    try:
        x = input()
        file=os.path.normpath(os.path.join(x, 'data.csv'))
        import pandas as pd
        read_data = pd.read_csv(file)
        return read_data
        pass
    except OSError:
        print("Path not valid. Please enter the path again")
        return readFile()
```

User is prompted for the file path. Give the path to the 'data.csv' file. If the path is found to be invalid, system will prompt the user for entering the path again and again until a correct valid path is found where 'data.csv' is kept.

2) In the main function, everything is inside try except block.
    a) Call the readFile() function.
    b) Filter Gender codes for males and females using 'Gender Code from Claim' column. Value '1' is for 'females' and value '2' is for 'males'. Also rename the columns of the data frame for females & males.
    c) Filter Age Category for ages (<65, 65-74, 75+) using 'LDS Age Category' column. Value 1 is for 'Ages < 65', Value [2, 3] is for 'Ages 65-74', Value [4, 5,6] is for 'Ages 75+'. Also rename the columns of the data frame for 'age_lessthan_65', 'age_between_65to74' & 'age_greaterthan_75'.
    d) Merge all the data frames: females, males, age_lessthan_65, age_between_65to74, age_greaterthan_75 and fill the NaN with 0's.

```
#Merge all the dataframes on column 'State'

final_output=pd.merge(pd.merge(pd.merge(pd.merge(females, males, on='State', how='outer'),age_lessthan_65,on='State', how='outer'),age_between_65to74,on='State', how='outer'),age_greaterthan_75,on='State', how='outer
```

    e) Rearrange the columns & write the final output to 'output1.csv' file. If the output file is already present in the current working directory, replace the file.

```
# Handle the exception if file with the same name as 'output2' is already present in the directory. Replace the file with new file
try:
    final_output[cols].to_csv('output1.csv',index = False)
except OSError:
    pass
    os.remove(filename)
    final_output[cols].to_csv('output1.csv',index = False)
```

Question 2:

1) User is prompted for the file path. Give the path to the 'data.csv' file. If the path is found to be invalid, system will prompt the user for entering the path again and again until a correct valid path is found where 'data.csv' is kept.
2) Group the data by 'Claim Utilization Day Count' to sum the number of claims for every number of Claim Utilization day.

```
import pandas as pd
# 1. Group the data by Claim utilization Day count
# 2. Count the number of claims for each category of claim utilization day where  categories are made when Claim utilization days are same.
# 3. Name the dataframe as 'total_claims_for_eachnumberof_utilization_day' which contains count of claims for each category of claim utilization day
total_claims_for_eachnumberof_utilization_day = pd.DataFrame(read_data.groupby('Claim Utilization Day Count')['Claim number'].count())
total_claims_for_eachnumberof_utilization_day['Utilization Range'] = total_claims_for_eachnumberof_utilization_day.index
total_claims_for_eachnumberof_utilization_day.columns = ['Counts','Utilization Range']
```

3) Make a list of all the unique number of Claim Utilization Days. List is 'all_utilization_day_ranges'. Append each unique day for Claim Utilization day in this list.
4) Calculate the sum of all the claims over *all the 'Claim Utilization days'*. This is required for calculating the percentages of claim count per every unique Claim Utilization day.(total_sum_of_all_claims)
5) Calculate the sum of all the claims over *every 'Claim Utilization day'*. This is required for calculating the percentages of claim count per every unique Claim Utilization day.(sum_of_claims_over_each_utilization_range)
6) Calculate the percentages of claims in each category range of Claim Utilization Day. Add them to the list 'percentage_list'.
7) Make the final data frame (final_frame)

```
# Make a dataframe which contains two Claim Utilization Day Range, Count of claims for each Utilization Day Range.
# Percentage of Claims for each range
final_frame=pd.DataFrame({'Utilization Range':["0","1","2","3","4","5","6-10","11-30","30 Plus"],'Counts':sum_of_claims_over_each_utilization_range,'Percentages':percentage_list})
```

8) Write the final data frame to 'output.csv' file.

Glossary:

Question 1:

file: *path to the data file*
read_data: *data frame containing the read 'data.csv' file.*
females: *data frame containing the count of females over each state code.*
males: *data frame containing the count of males over each state code.*
age_lessthan_65: *data frame containing the count of people with age<65 over each state code.*
age_between_65to74: *data frame containing the count of people with age (65-74) over each state code.*
age_greaterthan_75: *data frame containing the count of people with age>65 over each state code.*
final_output: *data frame for final output*
final_output[cols]: *data frame for final output with arranged order of columns.*
output1.csv: *final output file*

Question 2:

total_claims_for_eachnumberof_utilization_day: *the data frame containing count of claims over each Claim utilization day range. Claim Utilization day ranges are initially from 1-150. 150 is the maximum number of Claim Utilization day.*
all_utilization_day_ranges: *list containing all the claim utilization day ranges. Example [1,2, 3….150]*
total_sum_of_all_claims: *sum of all the claims over all Claim Utilization days ranges.*
sum_of_claims_over_each_utilization_range: *Sum of claims over every Individual claim utilization day range.*
range6_10: *list containing sequence 6 to 10 for the purpose of making a claim utilization range from 6-10.*
range11_30: *list containing sequence 11 to 30 for the purpose of making a claim utilization range from 11-30.*
range30_plus: *list containing sequence 6 to 10 for the purpose of making a claim utilization range for 30 plus days.*
all_ranges: *list containing all the multi day ranges. [6-10], [11-30],[30 plus].*
percentage_list: *list containing the percentages of count of claims for each Claim Utilization day range.*
final_frame: *data frame for final output.*
final_frame[final_columns]: *data frame for final output with arranged order of columns.*
output2.csv: *final output file.*