import time

import pandas as pd

import numpy as np

CITY\_DATA = { 'chicago': 'chicago.csv',

'new york city': 'new\_york\_city.csv',

'washington': 'washington.csv' }

def get\_filters():

"""

Asks user to specify a city, month, and day to analyze.

Returns:

(str) city - name of the city to analyze

(str) month - name of the month to filter by, or "all" to apply no month filter

(str) day - name of the day of week to filter by, or "all" to apply no day filter

"""

print('Hello! Let\'s explore some US bikeshare data!')

city = input('ENTER THE CITY: ').lower()

while city not in ['chicago', 'new york city', 'washington']:

city = input("CHOOSE chicago, new york city OR washington: ").lower()

# TO DO: get user input for month (all, january, february, ... , june)

month = input('ENTER MONTH: ').lower()

while month not in ['january', 'february', 'march', 'april', 'may', 'june', 'all']:

month = input('ENTER MONTH january, february, ... , june : ').lower()

# TO DO: get user input for day of week (all, monday, tuesday, ... sunday)

day = input('ENTER DAY : ').lower()

print('\_' \* 40)

return city, month, day

def load\_data(city, month, day):

"""

Loads data for the specified city and filters by month and day if applicable.

Args:

(str) city - name of the city to analyze

(str) month - name of the month to filter by, or "all" to apply no month filter

(str) day - name of the day of week to filter by, or "all" to apply no day filter

Returns:

df - Pandas DataFrame containing city data filtered by month and day

"""

# load data file into a dataframe

df = pd.read\_csv(CITY\_DATA[city])

# convert the Start Time column to datetime

df['Start Time'] = pd.to\_datetime(df['Start Time'])

# extract month and day of week from Start Time to create new columns

df['month'] = df['Start Time'].dt.month

df['day\_of\_week'] = df['Start Time'].dt.weekday\_name

# filter by month if applicable

if month != 'all':

# use the index of the months list to get the corresponding int

months = ['January', 'February', 'March', 'April', 'May', 'June']

month = next((index + 1 for index, m in enumerate(months) if m.lower() == month.lower()), None)

# filter by month to create the new dataframe

df = df[df['month'] == month]

# filter by day of week if applicable

if day != 'all':

# filter by day of week to create the new dataframe

df = df[df['day\_of\_week'] == day.title()]

return df

def time\_stats(df):

"""Displays statistics on the most frequent times of travel."""

print('\nCalculating The Most Frequent Times of Travel...\n')

start\_time = time.time()

# TO DO: display the most common month

popular\_month = df['month'].mode()[0]

print('Most Common Month:', popular\_month)

# TO DO: display the most common day of week

popular\_day = df['day\_of\_week'].mode()[0]

print('Most Common day:', popular\_day)

# TO DO: display the most common start hour

df['hour'] = df['Start Time'].dt.hour

popular\_hour = df['hour'].mode()[0]

print('Most Common Hour:', popular\_hour)

print("\nThis took %s seconds." % (time.time() - start\_time))

print('-'\*40)

def station\_stats(df):

"""Displays statistics on the most popular stations and trip."""

print('\nCalculating The Most Popular Stations and Trip...\n')

start\_time = time.time()

# TO DO: display most commonly used start station

print("most common start station: ", df ['Start Station'].value\_counts().idxmax())

# TO DO: display most commonly used end station

print("most common end station: ", df['End Station'].value\_counts().idxmax())

# TO DO: display most frequent combination of start station and end station trip

print("most frequent combination of start station and end station trip")

most\_common\_start\_and\_end\_stations = df.groupby(['Start Station', 'End Station']).size().nlargest(1)

print(most\_common\_start\_and\_end\_stations)

print("\nThis took %s seconds." % (time.time() - start\_time))

print('-'\*40)

def trip\_duration\_stats(df):

"""Displays statistics on the total and average trip duration."""

print('\nCalculating Trip Duration...\n')

start\_time = time.time()

# TO DO: display total travel time

Total\_Travel\_Time = sum(df['Trip Duration'])

print('Total travel time:', Total\_Travel\_Time/86400, " Days")

# TO DO: display mean travel time

Mean\_Travel\_Time = df['Trip Duration'].mean()

print('Mean travel time:', Mean\_Travel\_Time/60, " Minutes")

print("\nThis took %s seconds." % (time.time() - start\_time))

print('-'\*40)

def user\_stats(df):

"""Displays statistics on bikeshare users."""

print('\nCalculating User Stats...\n')

start\_time = time.time()

# TO DO: Display counts of user types

user\_types = df['User Type'].value\_counts()

print('User Types:\n', user\_types)

# TO DO: Display counts of gender

try:

gender\_types = df['Gender'].value\_counts()

print('\nGender Types:\n', gender\_types)

except KeyError:

print("\nGender Types:\nNo data available for this month.")

# TO DO: Display earliest, most recent, and most common year of birth

try:

Earliest\_Year = df['Birth Year'].min()

print('\nEarliest Year:', Earliest\_Year)

except KeyError:

print("\nEarliest Year:\nNo data available for this month.")

try:

Most\_Recent\_Year = df['Birth Year'].max()

print('\nMost Recent Year:', Most\_Recent\_Year)

except KeyError:

print("\nMost Recent Year:\nNo data available for this month.")

try:

Most\_Common\_Year = df['Birth Year'].value\_counts().idxmax()

print('\nMost Common Year:', Most\_Common\_Year)

except KeyError:

print("\nMost Common Year:\nNo data available for this month.")

print("\nThis took %s seconds." % (time.time() - start\_time))

print('-'\*40)

def display\_data(df):

"""Displays 5 lines of raw data at the time when yes is selected."""

d = 1

while True:

displaydata = input('\nWould you like to see 5 lines of data displayed? Enter yes or no. \n')

if displaydata.lower() == 'yes':

print(df[d:d+5])

d = d+5

else:

break

def main():

while True:

city, month, day = get\_filters()

df = load\_data(city, month, day)

time\_stats(df)

station\_stats(df)

trip\_duration\_stats(df)

user\_stats(df)

display\_data(df)

restart = input('\nWould you like to restart? Enter yes or no.\n')

if restart.lower() != 'yes':

break

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

main()