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

"""

valid\_cities = ['chicago', 'new york city', 'washington']

valid\_months = ['january', 'february', 'march', 'april', 'may', 'june', 'all']

valid\_days = ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday', 'all']

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

# TO DO: get user input for city (chicago, new york city, washington). HINT: Use a while loop to handle invalid inputs

while True:

city = input("Please enter the name of the city (Chicago, New York City, Washington): ").lower()

if city in valid\_cities:

break

else:

print("Invalid input. Please try again.")

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

while True:

month = input("Please enter the name of the month (January, February, ..., June, or 'all' for all months): ").lower()

if month in valid\_months:

break

else:

print("Invalid input. Please try again.")

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

while True:

day = input("Please enter the name of the day of the week (Monday, Tuesday, ..., Sunday, or 'all' for all days): ").lower()

if day in valid\_days:

break

else:

print("Invalid input. Please try again.")

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

"""

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

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

df['Month'] = df['Start Time'].dt.month\_name()

df['Day of Week'] = df['Start Time'].dt.day\_name()

#Filter by month only

if month != 'all' and day == 'all':

mdf = df[df['Month'] == month.title()]

return mdf

#Filter by day only

elif day != 'all' and month == 'all':

ddf = df[df['Day of Week'] == day.title()]

return ddf

#Filter by month and day

elif month != 'all' and day != 'all':

mddf = df[(df['Month'] == month.title()) & (df['Day of Week'] == day.title())]

return mddf

#No filter

else:

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

common\_month = df['Month'].value\_counts().idxmax()

print("The most common month is:", common\_month)

# TO DO: display the most common day of week

common\_day = df['Day of Week'].value\_counts().idxmax()

print("The most common day of the week is:", common\_day)

# TO DO: display the most common start hour

df['start\_hour'] = df['Start Time'].dt.hour

common\_start\_hour = df['start\_hour'].value\_counts().idxmax()

print("The most common start hour is:", common\_start\_hour)

def station\_stats(df):

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

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

# TO DO: display most commonly used start station

common\_start\_station = df['Start Station'].value\_counts().idxmax()

print("The most commonly used start station is:", common\_start\_station)

# TO DO: display most commonly used end station

common\_end\_station = df['End Station'].value\_counts().idxmax()

print("The most commonly used end station is:", common\_end\_station)

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

frequent\_combination = df.groupby(['Start Station', 'End Station']).size().idxmax()

print("The most frequent combination of start station and end station is:", frequent\_combination)

def trip\_duration\_stats(df):

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

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

# TO DO: display total travel time

total\_travel\_time = df['Trip Duration'].sum()

print("The total travel time is:", "{:.2f}".format(total\_travel\_time/60/60/24), "days.")

# TO DO: display mean travel time

mean\_travel\_time = df['Trip Duration'].mean()

print("The mean travel time is:", "{:.2f}".format(mean\_travel\_time/60), "minutes.")

def user\_stats(df):

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

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

# TO DO: Display counts of user types

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

print('Count of each user type:')

for k, v in user\_types.items():

print('{:10} : {:,}'.format(k, v))

# TO DO: Display counts of gender

if 'Gender' in df.columns: #washington.csv does not have a gender column

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

print('\nCount of each gender:')

for k, v in gender\_counts.items():

print('{:7} : {:,}'.format(k, v))

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

if 'Birth Year' in df.columns: #washington.csv does not have a birth year column

earliest\_year = df['Birth Year'].min()

most\_recent\_year = df['Birth Year'].max()

most\_common\_year = df['Birth Year'].mode().values[0]

print("\nThe earliest year of birth is:", int(earliest\_year))

print("The most recent year of birth is:", int(most\_recent\_year))

print("The most common year of birth is:", int(most\_common\_year))

def display\_raw\_data(df):

"""Displays raw data of the city dataframe based on consumer's demand."""

row\_index = 0

while True:

display = input("\nDo you want to see 5 lines of raw data? Enter 'yes' or 'no': ").lower()

if display == 'yes':

print(df.iloc[row\_index:row\_index+5])

row\_index += 5

elif display == 'no':

break

else:

print("Invalid input. Please enter 'yes' or 'no'.")

def display\_raw\_data(df):

"""Displays raw data of the city dataframe based on consumer's demand."""

row\_index = 0

while True:

display = input("\nDo you want to see 5 lines of raw data? Enter 'yes' or 'no': ").lower()

if display == 'yes':

print(df.iloc[row\_index:row\_index+5])

row\_index += 5

elif display == 'no':

break

else:

print("Invalid input. Please enter 'yes' or 'no'.")

def main():

"""Main function controlling flow of program"""

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\_raw\_data(df)

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

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

break

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

main()