Assignent_Mongo

June 20, 2020

[1]:
```
import datetime as datetime
import pandas as pd
import numpy as np
import pymongo
import re
import locale
import pdb
locale.setlocale(locale.LC_ALL, "nl_NL") # For conversion of dutch dates i.e␣
 ↪mei to May
```

[1]:
```
'nl_NL'
```

[ ]:
```
data = pd.read_csv('hue_upload.csv', header=None, delimiter=";")
allowedEvents =␣
 ↪['lamp_chang','nudge_time','bedtime_tonight','risetime','rise_reason','adherence␣
 ↪importance','fitness']
data.columns = ['row_id','user_id','event_id','value']
data = data[data.event_id.str.contains('|'.join(allowedEvents))] # Filter␣
 ↪allowed events
data['value'] = data['value'].str.replace(':','')
data['value']= data['value'].fillna('000000')
```

[2]:
```
def make_date(event):
    lamp_change_regex = '(_\d{1,4}_\w{1,10}_\d{4})'
    dateTime = re.findall(lamp_change_regex, event)[0]
    dateTime = dateTime[1:len(dateTime)]
    dateTime = pd.to_datetime(dateTime, format='%d_%B_%Y')
    return(dateTime)
```

[3]:
```
col_names =  ['date', 'user',␣
 ↪'bedtime','intended','risetime','rise_reason','fitness','adh','in_exp','sleep_duration']
df  = pd.DataFrame(columns = col_names)
```

[4]:
```
def update_bedtime(user,event,value):
    global df
    lamp_time_regex = '(_\d{1,2}_\d{1,2}_\d{1,2}_\d{1,1000})'
    date = make_date(event)
```

```
    time = re.findall(lamp_time_regex, event)[0]
    time = time[1:len(time)]
    time = pd.to_datetime(time, format='%H_%M_%S_%f').time()
    #pdb.set_trace()
    if value == 'OFF':
        filteredUser = df[(df['user']==user)]
        filteredDate = filteredUser[(filteredUser['date']>= date) &␣
↪(filteredUser['date']<= (date+datetime.timedelta(days=1)))]

        if filteredDate.shape[0]== 0: #No such data
            #insert
            if time >= datetime.time(19,0):

                df = df.append([{'date':date,'user':user,'bedtime':
↪time,'in_exp':False}], ignore_index=True)
        else:
            try:
                if ((filteredDate['bedtime']>=datetime.time(19,0))).any():
                    maxTime = np.nanmax(filteredDate['bedtime'])
                    if time > maxTime:
                        df.loc[(df['user'] == user) & (df['date'] ==␣
↪date),['bedtime']]= time
                elif ((filteredDate['bedtime']<=datetime.time(6,0))).any():
                    minTime = np.nanmin(filteredDate['bedtime'])
                    if time < minTime:
                        df.loc[(df['user'] == user) & (df['date'] ==␣
↪date),['bedtime']]= time
                df.loc[(df['user'] == user) & (df['date'] == date),['in_exp']]=␣
↪False
            except:
                pass
```

[5]:
```
def update_intendedBedtime_riseTime_exp(user,event,value,var):
    exp = False
    if var =='in_exp':
        exp = True

    global df
    date = make_date(event)
    time_regex = '(\d{1,4})'

    time = re.findall(time_regex, value)[0]
    time = time.zfill(4)
    if time =='2400':
        time ='0000'

    try:
```

```
        time = pd.to_datetime(time,format='%H%M').time()
    except:
        time ='0000'
        time = pd.to_datetime(time,format='%H%M').time()
    filteredUser = df[(df['user']==user)]
    filteredDate = filteredUser[(filteredUser['date']>= date)]

    if filteredDate.shape[0]== 0:

        df = df.append([{'date':date,'user':user,var:time,'in_exp':exp}],␣
↪ignore_index=True)
    else:
        df.loc[(df['user'] == user) & (df['date'] == date),var]= time
        df.loc[(df['user'] == user) & (df['date'] == date),'in_exp']= exp
```

[6]:
```
def update_riseReason_fitness_adh(user,event,value,var):
    global df
    date = make_date(event)
    time = np.nan
    filteredUser = df[(df['user']==user)]
    filteredDate = filteredUser[(filteredUser['date']>= date)]

    if filteredDate.shape[0]== 0:
        df = df.append([{'date':date,'user':user,var:time,'in_exp':False}],␣
↪ignore_index=True)
    else:
        df.loc[(df['user'] == user) & (df['date'] == date),var]= value
```

[7]:
```
def update_exp(user,event,value,var):
    global df
    date = make_date(event)
    time = re.findall(time_regex, value)[0]

    if time =='2400':
        time ='0000'
    try:
        time = pd.to_datetime(time,format='%H%M').time()
    except:
        time = '0000'
        time = pd.to_datetime(time,format='%H%M').time()
    filteredUser = df[(df['user']==user)]
    filteredDate = filteredUser[(filteredUser['date']>= date)]

    if filteredDate.shape[0]== 0:
        df = df.append([{'date':date,'user':user,var:time,'in_exp':True}],␣
↪ignore_index=True)
```

```
    else:
        df.loc[(df['user'] == user) & (df['date'] == date),var]= value
```

[8]:
```
def read_csv_data1(filenames):

    for files in filenames:
        data = pd.read_csv(files, header=None, delimiter=";")

        allowedEvents =␣
↪['lamp_chang','nudge_time','bedtime_tonight','risetime','rise_reason','adherence␣
↪importance','fitness']
        data.columns = ['row_id','user_id','event_id','value']
        data = data[data.event_id.str.contains('|'.join(allowedEvents))] #␣
↪Filter allowed events
        data['value'] = data['value'].str.replace(':','')
        data['value']= data['value'].fillna('000000')
        data = data.reset_index()

        print('Processing File: ',files)
        for index, row in data.iterrows():

            if 'lamp' in row.event_id:
                update_bedtime(row.user_id,row.event_id,row.value)
            if 'bedtime' in row.event_id:
                update_intendedBedtime_riseTime_exp(row.user_id,row.
↪event_id,row.value,'intended')
            if 'risetime' in row.event_id:
                update_intendedBedtime_riseTime_exp(row.user_id,row.
↪event_id,row.value,'risetime')
            if 'rise_reason' in row.event_id:
                update_riseReason_fitness_adh(row.user_id,row.event_id,row.
↪value,'rise_reason')
            if 'fitness' in row.event_id:
                update_riseReason_fitness_adh(row.user_id,row.event_id,row.
↪value,'fitness')
            if 'adherence_importance' in row.event_id:
                update_riseReason_fitness_adh(row.user_id,row.event_id,row.
↪value,'adherence_importance')
            if 'nudge_time' in row.event_id:
                update_intendedBedtime_riseTime_exp(row.user_id,row.
↪event_id,row.value,'in_exp')
```

[9]:
```
read_csv_data1(['hue_upload.csv','hue_upload2.csv'])
```

```
Processing File:  hue_upload.csv
Processing File:  hue_upload2.csv
```

[12]:
```
df.set_index(['user', 'date'], inplace=True)
df['bedtime'] = pd.to_datetime(df['bedtime'],format="%H:%M:%S.
 ↪%f",errors='coerce')
df['risetime'] = pd.to_datetime(df['risetime'],format="%H:%M:
 ↪%S",errors='coerce')
df['intended'] = pd.to_datetime(df['intended'],format="%H:%M:
 ↪%S",errors='coerce')
df['sleep_duration'] = (df.bedtime-df.risetime).astype('timedelta64[s]')
df["sleep_duration"] = df["sleep_duration"].astype(float)
df["sleep_duration"] = df["sleep_duration"].fillna(0)
df['bedtime'] = df['bedtime'].astype('datetime64[ns]')
df['risetime'] = df['risetime'].astype('datetime64[ns]')
df['intended'] = df['intended'].astype('datetime64[ns]')
df.fillna("-",inplace=True)
```

[13]:
```
# check the index
print(df.index)
```

```
MultiIndex([(10, '2015-05-29'),
            (10, '2015-05-31'),
            (37, '2015-05-28'),
            (37, '2015-05-31'),
            (12, '2015-06-01'),
            (12, '2015-06-06'),
            (24, '2015-05-30'),
            (24, '2015-05-31'),
            ( 1, '2015-05-31'),
            (20, '2015-05-29'),
            …
            (61, '2015-09-17'),
            (58, '2015-09-17'),
            (55, '2015-09-18'),
            (58, '2015-09-18'),
            (63, '2015-09-18'),
            (61, '2015-09-18'),
            (61, '2015-09-19'),
            (55, '2015-09-19'),
            (58, '2015-09-19'),
            (63, '2015-09-19')],
           names=['user', 'date'], length=408)
```

[14]:
```
# check the bedtime column
display(df[['bedtime']])
```

		bedtime
user	date	
10	2015-05-29	1900-01-01 19:08:33.984000

```
     2015-05-31                            -
37   2015-05-28  1900-01-01 21:45:42.339000
     2015-05-31                            -
12   2015-06-01                            -
...                                      ...
61   2015-09-18                            -
     2015-09-19                            -
55   2015-09-19                            -
58   2015-09-19                            -
63   2015-09-19                            -

[408 rows x 1 columns]
```

[15]:
```
# check the intended_bedtime column
display(df[['intended']])
```

```
                          intended
user date
10   2015-05-29                  -
     2015-05-31  1900-01-01 23:00:00
37   2015-05-28                  -
     2015-05-31  1900-01-01 12:00:00
12   2015-06-01  1900-01-01 00:00:00
...                             ...
61   2015-09-18  1900-01-01 23:00:00
     2015-09-19  1900-01-01 23:00:00
55   2015-09-19  1900-01-01 00:30:00
58   2015-09-19  1900-01-01 23:00:00
63   2015-09-19  1900-01-01 00:00:00

[408 rows x 1 columns]
```

[16]:
```
# check the rise_time
display(df[['risetime']])
```

```
                          risetime
user date
10   2015-05-29                  -
     2015-05-31  1900-01-01 09:00:00
37   2015-05-28                  -
     2015-05-31  1900-01-01 08:30:00
12   2015-06-01  1900-01-01 09:30:00
...                             ...
61   2015-09-18  1900-01-01 09:00:00
     2015-09-19  1900-01-01 11:00:00
55   2015-09-19  1900-01-01 11:00:00
```

```
58   2015-09-19  1900-01-01 07:00:00
63   2015-09-19  1900-01-01 09:45:00

[408 rows x 1 columns]
```

[17]:
```
# check the rise_reason, fitness, adherence_importance column
display(df[['rise_reason', 'fitness', 'adh']])
```

```
                rise_reason fitness adh
user date
10   2015-05-29           -       -   -
     2015-05-31          ja      52   -
37   2015-05-28           -       -   -
     2015-05-31          ja      34   -
12   2015-06-01          ja      83   -
...                     ...     ...  ..
61   2015-09-18         nee      27   -
     2015-09-19         nee      19   -
55   2015-09-19         nee      30   -
58   2015-09-19         nee      33   -
63   2015-09-19         nee      54   -

[408 rows x 3 columns]
```

[18]:
```
# check the in_experimental_group column
display(df[['in_exp']])
```

```
                in_exp
user date
10   2015-05-29  False
     2015-05-31   True
37   2015-05-28  False
     2015-05-31  False
12   2015-06-01  False
...                ...
61   2015-09-18  False
     2015-09-19  False
55   2015-09-19  False
58   2015-09-19  False
63   2015-09-19  False

[408 rows x 1 columns]
```

[19]:
```
def to_mongodb(df):
    # Connect to MongoDB
    client =  pymongo.MongoClient("localhost", 27017)
```

```
    db = client['BigData']
    collection = db['SleepData']
    df.reset_index(inplace=True)

    try:
        df.drop(['index'], axis=1)
    except:
        pass
    df_dict = df.to_dict("records")
    # Insert collection
    collection.insert_many(df_dict)
```

[20]:
```
to_mongodb(df)
```

[21]:
```
def read_mongodb(filterQ,sortid):

    client =  pymongo.MongoClient("localhost", 27017)
    db = client['BigData']
    collection = db['SleepData']
    query = filterQ
    mydoc = collection.find(query).sort(sortid)
    dfb = pd.DataFrame(list(mydoc))
    dfb.bedtime = dfb.bedtime.dt.time
    dfb.intended = dfb.intended.dt.time
    dfb.risetime = dfb.risetime.dt.time

    try:
        dfb.drop(['_id'],axis = 1,inplace=True)

    except:
        pass
    return(dfb)
```

[22]:
```
query = read_mongodb({'sleep_duration': {'$gt': 50}}, '_id')
query
```

[22]:

	user	date	bedtime	intended	risetime	rise_reason	fitness	adh	in_exp	sleep_duration
0	20	2015-05-31	22:31:44.148000	00:23:00	08:50:00	nee	45			
1	12	2015-06-03	23:47:00.370000	23:30:00	09:00:00	ja	28			
2	12	2015-06-04	23:31:30.310000	23:30:00	07:30:00	ja	75			
3	32	2015-06-10	23:28:47.814000	23:00:00	11:00:00	ja	65			
4	39	2015-08-16	20:29:24.415000	00:00:00	11:01:00	ja	59			
5	42	2015-08-18	23:59:45.244000	00:00:00	06:20:00	ja	40			
6	60	2015-09-09	21:43:19.097000	00:26:00	06:38:00	ja	80			
7	63	2015-09-16	23:43:48.837000	23:30:00	08:00:00	ja	48			

```
0   -   False         49304.0
1   -   False         53220.0
2   -   False         57690.0
3   -   False         44927.0
4   -    True         34104.0
5   -   False         63585.0
6   -   False         54319.0
7   -   False         56628.0
```

[ ]: