**Python teste rezolvate**

**Testul 1**

În fișierul ***Etnicity.csv***se află date privind populația pe etnii la nivelul localităților conform Recensământului Populației și Locuințelor din 2011. În fișierul ***CoduriRomania.xlsx*** sunt codificările localităților, județelor, regiunilor și macroregiunilor României.

**1. Să se calculeze și să se salveze populația pe etnii la nivelul județelor, regiunilor și macroregiunilor**

**2. Să se calculeze și să se salveze indicii de segregare etnică la nivelul județelor.** Vor fi utilizați următorii indici: Indicele de disimilaritate,

D = 1 2 ∑ | 𝑥𝑖 𝑇𝑥 − 𝑟𝑖 𝑇𝑟 | 𝑛 𝑖=1

Shannon-Weaver, H = − ∑ 𝑝𝑖 log2 𝑝𝑖 𝑛 𝑖=1 unde n este numărul de unități administrativ-teritoriale, xi este populația de o anumită etnie, ri este restul populației, Tx este totalul populației de etnia respectivă, Tr este totalul restului populației, pi este ponderea etniei la nivelul unității administrativ-teritoriale i.

**Rezolvare:**

*Main.py*

import pandas as pd  
import functii as f  
  
etnii = pd.read\_csv("Ethnicity.csv", index\_col=0)  
f.nan\_replace(etnii)  
# Citire coduri  
localitati = pd.read\_excel("CoduriRomania.xlsx", index\_col=0)  
judete = pd.read\_excel("CoduriRomania.xlsx", sheet\_name=1, index\_col=0)  
regiuni = pd.read\_excel("CoduriRomania.xlsx",  
 sheet\_name="Regiuni", index\_col=0)  
  
variabile = list(etnii.columns)[1:]  
  
# Calcul populatie pe etnii la nivel de judet  
t1 = etnii.merge(right=localitati, left\_index=True, right\_index=True)  
variabile1 = variabile + ["County"]  
g1 = t1[variabile1].groupby(by="County").agg(sum)  
# print(g1)  
assert isinstance(g1, pd.DataFrame)  
g1.to\_csv("PopulatieJudete.csv")  
  
# Calcul populatie pe etnii la nivel de regiune  
t2 = g1.merge(right=judete, left\_index=True, right\_index=True)  
variabile2 = variabile + ["Regiune"]  
g2 = t2[variabile2].groupby(by="Regiune").agg(sum)  
g2.to\_csv("PopulatieRegiuni.csv")  
  
# Calcul populatie pe etnii la nivel de macroregiune  
assert isinstance(g2, pd.DataFrame)  
t3 = g2.merge(right=regiuni, left\_index=True, right\_index=True)  
variabile3 = variabile + ["MacroRegiune"]  
g3 = t3[variabile3].groupby(by="MacroRegiune").agg(sum)  
g3.to\_csv("PopulatieMacroregiuni.csv")  
  
# Calcul segregare la nivel de judet dupa indexul Shannon  
segregare\_Shannon = t1[variabile1].groupby(by="County").agg(  
 func = f.segregare\_S, v = variabile  
)  
segregare\_Shannon.to\_csv("SegregareShannon.csv")  
  
segregare\_Disim = t1[variabile1].groupby(by="County").agg(  
 func = f.segregare\_D, v = variabile  
)  
segregare\_Disim.to\_csv("SegregareDisimilaritate.csv")

*Functii.py*

import pandas as pd  
import pandas.api.types as tip  
from scipy.stats import entropy  
import numpy as np  
  
  
def nan\_replace(t):  
 assert isinstance(t, pd.DataFrame)  
 variabile = list(t.columns)  
 for variabila in variabile:  
 if any(t[variabila].isna()):  
 if tip.is\_numeric\_dtype(t[variabila]):  
 t[variabila].fillna(t[variabila].mean(), inplace=True)  
 else:  
 modulul = t[variabila].mode()[0]  
 t[variabila].fillna(modulul, inplace=True)  
  
  
def segregare\_S(t, v):  
 assert isinstance(t, pd.DataFrame)  
 x = t[v].values  
 sume = np.sum(x, axis=0)  
 sume[sume == 0] = 1  
 p = x / sume  
 p[p == 0] = 1  
 e = entropy(p, base=2, axis=0)  
 s\_e = pd.Series(data=e, index=v)  
 return s\_e  
  
  
def segregare\_D(t, v):  
 assert isinstance(t, pd.DataFrame)  
 x = t[v].values  
 sume\_linii = np.sum(x, axis=1)  
 r = np.transpose(sume\_linii - x.T)  
 t\_x = np.sum(x, axis=0)  
 t\_r = np.sum(r, axis=0)  
 t\_x[t\_x == 0] = 1  
 t\_r[t\_r == 0] = 1  
 p\_x = x / t\_x  
 p\_r = r / t\_r  
 dif = np.abs(p\_x - p\_r)  
 sum\_dif = 0.5 \* np.sum(dif, axis=0)  
 return pd.Series(data=sum\_dif, index=v)

**Testul 2**

În fișierul ***Etnicity.csv*** se află date privind populația pe etnii la nivelul localităților conform Recensământului Populației și Locuințelor din 2011. În fișierul ***CoduriRomania.xlsx*** sunt codificările localităților, județelor, regiunilor și macroregiunilor României.

**1. Să se calculeze și să se salveze populația pe etnii la nivelul județelor, regiunilor și macroregiunilor**

**2. Să se calculeze și să se salveze indicii de diversitate etnică la nivel de județ și regiune.** Vor fi folosite următoarele relații de calcul pentru diversitate: Shannon-Weaver, H = − ∑ 𝑝𝑖 log2 𝑝𝑖 𝑛 𝑖=1 Simpson, S1 = 1- ∑ 𝑝𝑖 𝑛 2 𝑖=1 Inverse Simpson, S2 = 1 ∑ 𝑝𝑖 𝑛 2 𝑖=1 unde n este numărul de etnii, pi este ponderea etniei i la nivelul unității administrativ-teritoriale.

**Rezolvare:**

*Main.py*

import pandas as pd  
import functii as f  
  
etnii = pd.read\_csv("Ethnicity.csv", index\_col=0)  
  
localitati = pd.read\_excel("CoduriRomania.xlsx", index\_col=0)  
judete = pd.read\_excel("CoduriRomania.xlsx", sheet\_name=1, index\_col=0)  
regiuni = pd.read\_excel("CoduriRomania.xlsx", sheet\_name="Regiuni", index\_col=0)  
  
f.nan\_replace(etnii)  
# print(etnii)  
  
variabile = list(etnii.columns)[1:]  
  
# Calcul populatie pe etnii la nivel de judet  
t1 = etnii.merge(right=localitati, left\_index=True, right\_index=True)  
variabile1 = variabile + ["County"]  
g1 = t1[variabile1].groupby(by="County").agg(sum)  
assert isinstance(g1, pd.DataFrame)  
g1.to\_csv("PopulatieEtniiJudete.csv")  
  
# Calcul populatie pe etnii la nivel de regiune  
t2 = g1.merge(right=judete, left\_index=True, right\_index=True)  
variabile2 = variabile + ["Regiune"]  
g2 = t2[variabile2].groupby(by="Regiune").agg(sum)  
g2.to\_csv("PopulatieEtniiRegiuni.csv")  
  
# Calcul populatie pe etnii la nivel de macroregiune  
assert isinstance(g2, pd.DataFrame)  
t3 = g2.merge(regiuni, left\_index=True, right\_index=True)  
variabile3 = variabile + ["MacroRegiune"]  
g3 = t3[variabile3].groupby(by="MacroRegiune").agg(sum)  
g3.to\_csv("PopulatieEtniiMacroregiuni.csv")  
  
# Calcul diversitate la nivel de localitate  
# print(list(t1))  
div1 = t1[variabile+["City\_x"]].apply(func=f.diversitate,axis=1)  
div1.to\_csv("DiversitateLocalitati.csv")  
  
# Calcul diversitate la nivel de judete  
div2 = g1[variabile].apply(func=f.diversitate\_,axis=1,variabile=variabile)  
div2.to\_csv("DiversitateJudete.csv")  
  
div3 = g2[variabile].apply(func=f.diversitate\_,axis=1,variabile=variabile)  
div3.to\_csv("DiversitateRegiuni.csv")

*Functii.py*

import numpy as np  
import pandas as pd  
from pandas.api.types import is\_numeric\_dtype  
from scipy.stats import entropy  
  
  
def nan\_replace(t):  
 assert isinstance(t, pd.DataFrame)  
 variabile = list(t.columns)  
 for variabila in variabile:  
 if any(t[variabila].isna()):  
 if is\_numeric\_dtype(t[variabila]):  
 t[variabila].fillna(t[variabila].mean(), inplace=True)  
 else:  
 modulul = t[variabila].mode()[0]  
 t[variabila].fillna(modulul, inplace=True)  
  
def diversitate\_(t,variabile):  
 assert isinstance(t,pd.Series)  
 x = t[variabile].values  
 s = np.sum(x)  
 p = x / s  
 p[p == 0] = 1  
 e = -np.sum(p \* np.log2(p))  
 s\_s = np.sum(p\*p)  
 simpson = 1 - s\_s  
 inv\_simpson = 1/s\_s  
 s\_div = pd.Series(  
 data= [e,simpson,inv\_simpson],  
 index=["Shannon","Simpson","Inverse Simpson"]  
 )  
 return s\_div  
  
  
def diversitate(t):  
 assert isinstance(t, pd.Series)  
 x = np.array(t.iloc[:-1],dtype=float)  
 s = np.sum(x)  
 p = x / s  
 p[p == 0] = 1  
 e = -np.sum(p \* np.log2(p))  
 s\_s = np.sum(p\*p)  
 simpson = 1 - s\_s  
 inv\_simpson = 1/s\_s  
 s\_div = pd.Series(  
 data= [t.iloc[len(t)-1],e,simpson,inv\_simpson],  
 index=["Localitate","Shannon","Simpson","Inverse Simpson"]  
 )  
 return s\_div

**Testul 2\_1**

*Main.py*

import pandas as pd  
  
import functii as f  
  
tabel\_etnii = pd.read\_csv("Ethnicity.csv", index\_col=0)  
f.nan\_replace(tabel\_etnii)  
# print(tabel\_etnii)  
variabile\_etnii = list(tabel\_etnii.columns)[1:]  
  
# Calcul populatie pe etnii la nivel de judet  
localitati = pd.read\_excel("CoduriRomania.xlsx", index\_col=0)  
t1 = tabel\_etnii.merge(right=localitati, left\_index=True, right\_index=True)  
# print(t1)  
g1 = t1[variabile\_etnii + ["County"]].groupby(by="County").agg(sum)  
assert isinstance(g1, pd.DataFrame)  
g1.to\_csv("EtniiJudete.csv")  
  
# Calcul populatie pe etnii la nivel de regiune  
judete = pd.read\_excel("CoduriRomania.xlsx", sheet\_name=1, index\_col=0)  
t2 = g1.merge(judete, left\_index=True, right\_index=True)  
g2 = t2[variabile\_etnii + ["Regiune"]].groupby(by="Regiune").agg(sum)  
assert isinstance(g2, pd.DataFrame)  
g2.to\_csv("EtniiRegiuni.csv")  
  
# Calcul populatie pe etnii la nivel de macroregiune  
regiuni = pd.read\_excel("CoduriRomania.xlsx", sheet\_name="Regiuni", index\_col=0)  
t3 = g2.merge(regiuni, left\_index=True, right\_index=True)  
g3 = t3[variabile\_etnii + ["MacroRegiune"]].groupby(by="MacroRegiune").agg(sum)  
assert isinstance(g3, pd.DataFrame)  
g3.to\_csv("EtniiMacroregiuni.csv")  
  
# Calcul indici de diversitate la nivel de localitate  
# div1 = tabel\_etnii.apply(func=f.diversitate,axis=1)  
# div1.to\_csv("DiversitateLocalitati.csv")  
div1 = tabel\_etnii.apply(func=f.diversitate\_\_,axis=1,  
 coloana\_denumire="Localitate")  
div1.to\_csv("DiversitateLocalitati.csv")  
  
  
# Calcul indici de diversitate la nivel de judet  
# div2 = g1.apply(func=f.diversitate\_,axis=1)  
# div2.to\_csv("DiversitateJudete.csv")  
div2 = g1.apply(func=f.diversitate\_\_,axis=1)  
div2.to\_csv("DiversitateJudete.csv")  
  
# Calcul indici de diversitate la nivel de regiune  
# div3 = g2.apply(func=f.diversitate\_,axis=1)  
# div3.to\_csv("DiversitateRegiuni.csv")  
div3 = g2.apply(func=f.diversitate\_\_,axis=1)  
div3.to\_csv("DiversitateRegiuni.csv")

*Functii.py*

import pandas as pd  
from pandas.api.types import is\_numeric\_dtype  
import numpy as np  
  
  
def nan\_replace(t):  
 assert isinstance(t, pd.DataFrame)  
 nume\_variabile = list(t.columns)  
 for v in nume\_variabile:  
 if any(t[v].isna()):  
 if is\_numeric\_dtype(t[v]):  
 t[v].fillna(t[v].mean(), inplace=True)  
 else:  
 modulul = t[v].mode()[0]  
 t[v].fillna(modulul, inplace=True)  
  
  
def diversitate\_\_(t, coloana\_denumire=None):  
 assert isinstance(t, pd.Series)  
 if coloana\_denumire is not None:  
 x = np.array(t.iloc[1:], dtype=float)  
 else:  
 x = t.values  
 s = np.sum(x)  
 p = x / s  
 k = p == 0  
 p[k] = 1  
 e = -np.sum(p \* np.log2(p))  
 p\_ = x / s  
 s\_p = np.sum(p\_ \* p\_)  
 simpson = 1 - s\_p  
 inv\_simpson = 1 / s\_p  
 if coloana\_denumire is not None:  
 serie\_div = pd.Series(  
 data=[t.iloc[0], e, simpson, inv\_simpson],  
 index=[coloana\_denumire, "Shannon", "Simpson", "Inverse Simpson"]  
 )  
 else:  
 serie\_div = pd.Series(  
 data=[e, simpson, inv\_simpson],  
 index=["Shannon", "Simpson", "Inverse Simpson"]  
 )  
 return serie\_div  
  
  
def diversitate(t):  
 assert isinstance(t, pd.Series)  
 x = np.array(t.iloc[1:], dtype=float)  
 s = np.sum(x)  
 p = x / s  
 k = p == 0  
 p[k] = 1  
 e = -np.sum(p \* np.log2(p))  
 p\_ = x / s  
 s\_p = np.sum(p\_ \* p\_)  
 simpson = 1 - s\_p  
 inv\_simpson = 1 / s\_p  
 serie\_div = pd.Series(  
 data=[t.iloc[0], e, simpson, inv\_simpson],  
 index=["Localitate", "Shannon", "Simpson", "Inverse Simpson"]  
 )  
 return serie\_div  
  
  
def diversitate\_(t):  
 assert isinstance(t, pd.Series)  
 x = t.values  
 s = np.sum(x)  
 p = x / s  
 k = p == 0  
 p[k] = 1  
 e = -np.sum(p \* np.log2(p))  
 p\_ = x / s  
 s\_p = np.sum(p\_ \* p\_)  
 simpson = 1 - s\_p  
 inv\_simpson = 1 / s\_p  
 serie\_div = pd.Series(  
 data=[e, simpson, inv\_simpson],  
 index=["Shannon", "Simpson", "Inverse Simpson"]  
 )  
 return serie\_div

**Testul 3**

În fișierul ***prezenta\_vot.csv*** sunt prezentate date privind prezența la vot la nivelul localităților la alegerile parlamentare din 2020. Câmpurile sunt următoarele: Siruta - Cod siruta, Localitate - Nume localitate, Judet - Indicativ județ, Mediu - Mediul urban sau rural, Nr\_sectie - Număr secție, Votanti\_LP - Număr votanți înscriși lista permanentă, Votanti\_LS - Votanți inscriși pe lista suplimentară, LP - Voturi pe lista permanentă, LS - Voturi pe lista suplimentară, LSC - Voturi prin corespondență, UM - Voturi cu urna mobilă, LT - Total voturi, Barbati\_18-24, Barbati\_25-34, ..., Femei\_65\_ - Voturi pe categorii de vârstă și sexe. În fișierul ***CoduriRomania.xlsx*** sunt codificările localităților, județelor, regiunilor și macroregiunilor României conform NUTS (Nomenclatorul Unităților Teritoriale pentru Statistică din România).

Să se scrie o aplicație care să efectueze următoarele operațiuni:

**1. Salvarea în fișierul *Prezenta50.csv* a localităților în care prezența la vot a fost mai mare decât 50%. 2. Salvarea în fișierul *PrezentaSort.csv* a localităților sortate descrescător după procentul de participare la vot.**

**3. Salvarea în fișierul *Regiuni.csv* a prezenței la vot la nivelul regiunilor.**

**4. Determinarea și salvarea în fișierul *Varsta.csv* pentru fiecare localitate a categoriei de vârstă cu cea mai mare prezență la vot.**

**5. Să se salveze în fișierul *categorie\_varsta.csv*, localitățile în care categorie\_varsta a avut cea mai mare prezență la vot.**

**Indicație Relația de calcul pentru procentul participării la vot este: LT\*100/(Votanti\_LP+ Votanti\_LS)**

**Rezolvare:**

*Main.py*

import pandas as pd  
import functii as f  
import numpy as np  
  
prezenta = pd.read\_csv("prezenta\_vot.csv", index\_col=0)  
f.nan\_replace(prezenta)  
# print(prezenta)  
variabile = list(prezenta)  
# print(variabile)  
variabile1 = variabile[3:10]  
variabile2 = variabile[10:]  
# print(variabile1,variabile2,sep="\n")  
  
# Cerinta 1  
procent = prezenta["LT"] \* 100 / (prezenta["Votanti\_LP"] + prezenta["Votanti\_LS"])  
t1 = prezenta[procent > 50]  
t1.to\_csv("Prezenta50.csv")  
  
# Cerinta 2  
prezenta["procent"] = procent  
t2 = prezenta[["Localitate", "Judet", "procent"]].sort\_values(by="procent", ascending=False)  
t2.to\_csv("PrezentaSort.csv")  
  
# Cerinta 2. Versiunea 2  
k = np.flipud(np.argsort(procent))  
# print(k)  
t2\_ = prezenta[["Localitate", "Judet", "procent"]].iloc[k, :]  
t2\_.to\_csv("PrezentaSort\_.csv")  
  
# Cerinta 3  
judete = pd.read\_excel("CoduriRomania.xlsx", sheet\_name="Judete", index\_col=0)  
assert isinstance(judete, pd.DataFrame)  
prezenta\_complet = judete.merge(  
 right=prezenta,  
 left\_index=True,  
 right\_on="Judet"  
)  
t3 = prezenta\_complet[variabile1 + variabile2 + ["Regiune"]].groupby(by="Regiune").agg(sum)  
t3.to\_csv("Regiuni.csv")  
  
# Cerinta 4  
t4 = prezenta[["Localitate"] + variabile2].apply(func=f.vot\_categorii, axis=1)  
t4.to\_csv("Varsta.csv")  
  
# Cerinta 5  
categorie\_varsta = "Femei\_65\_"  
t5 = t4[t4["Categoria"]==categorie\_varsta]  
t5.to\_csv(categorie\_varsta+".csv")

*Functii.py*

import pandas as pd  
from pandas.api.types import is\_numeric\_dtype  
import numpy as np  
  
  
def nan\_replace(t):  
 assert isinstance(t, pd.DataFrame)  
 variabile = list(t.columns)  
 for variabila in variabile:  
 if any(t[variabila].isna()):  
 if is\_numeric\_dtype(t[variabila]):  
 t[variabila].fillna(t[variabila].mean(), inplace=True)  
 else:  
 modulul = t[variabila].mode()[0]  
 t[variabila].fillna(modulul, inplace=True)  
  
  
def vot\_categorii(t):  
 assert isinstance(t, pd.Series)  
 k = np.argmax(t.iloc[1:]) + 1  
 s = pd.Series(data={  
 "Localitatea": t.iloc[0],  
 "Categoria": t.index[k]  
 })  
 return s

**Testul 4**

În fișierul ***vot\_senat.csv*** sunt prezentate date privind votul pentru Senatul României la nivelul localităților la alegerile parlamentare din 2020. Câmpurile sunt următoarele: uat\_siruta - Cod Siruta localitate; nr\_alegatori - Număr alegători pe liste; nr\_voturi - Număr voturi valabil exprimate; PMP,PSD,PNL,USR\_PLUS,PRO\_ROMANIA,AUR,UDMR - Număr voturi acordate principalelor partide. În fișierul ***CoduriRomania.xlsx*** sunt codificările localităților, județelor, regiunilor și macroregiunilor României conform NUTS (Nomenclatorul Unităților Teritoriale pentru Statistică din România).

Să se scrie o aplicație care să efectueze următoarele operațiuni:

**1. Salvarea în fișierul *vot\_procente.csv* a rezultatelor votului pe partide, la nivelul localităților, exprimat în procente pe fiecare partid. De exemplu pentru PMP procentul este PMP\*100/nr\_voturi.**

**2. Salvarea în fișierul *PrezentaVot.csv* a localităților sortate descrescător după procentul de participare la vot. Se va salva codul siruta, numele localității și procentul participării la vot.**

**3. Salvarea în fișierul *Judete.csv* a rezultatelor votului pe partide la nivel de județ.**

**4. Determinarea și salvarea în fișierul *VotMajoritar.csv* pentru fiecare localitate a partidului cu cel mai mare procent de voturi obținute. Se va afișa codul siruta, numele localității și partidul majoritar.**

**5. Să se salveze în fișierul *nume\_partid.csv* lista localităților în care nume\_partid a câștigat alegerile.** I\

ndicație Relația de calcul pentru procentul participării la vot este: nr\_voturi\*100/nr\_alegatori

**Rezolvare:**

*Main.py*

import pandas as pd  
import functii as f  
  
vot = pd.read\_csv("vot\_senat.csv", index\_col=0)  
f.nan\_replace(vot)  
# print(vot)  
variabile = list(vot.columns)  
partide = variabile[2:]  
  
# Cerinta 1  
t1 = vot.apply(func=f.calcul\_procente, axis=1,  
 variabile=partide,  
 variabila\_raport="nr\_voturi")  
# print(t1)  
t1.to\_csv("vot\_procente.csv")  
  
# Cerinta 2  
vot["prezenta\_la\_vot"] = vot["nr\_voturi"] \* 100 / vot["nr\_alegatori"]  
localitati = pd.read\_excel("CoduriRomania.xlsx", index\_col=0)  
t2 = vot.merge(right=localitati, left\_index=True, right\_index=True)  
t2\_ = t2.sort\_values(by="prezenta\_la\_vot", ascending=False)  
t2\_.index.name = "Cod Siruta"  
t2\_[["City", "prezenta\_la\_vot"]].to\_csv("PrezentaVot.csv")  
  
# Cerinta 3  
t3 = t2.groupby(by="County").agg(sum)  
t3[partide].to\_csv("Judete.csv")  
  
# Cerinta 4  
t4 = t2[ ["City"] + partide ].\  
 apply(func=f.vot\_majoritar,axis=1)  
t4.index.name="Cod Siruta"  
# print(t4)  
t4.to\_csv("VotMajoritar.csv")  
  
# Cerinta 5  
nume\_partid = "UDMR"  
t5 = t4[ t4["Partid majoritar"]==nume\_partid ]  
# print(t5)  
t5.to\_csv(nume\_partid+".csv")

*Functii.py*

import pandas as pd  
from pandas.api.types import is\_numeric\_dtype  
import numpy as np  
  
  
def nan\_replace(t):  
 assert isinstance(t, pd.DataFrame)  
 variabile = list(t.columns)  
 for variabila in variabile:  
 if any(t[variabila].isna()):  
 if is\_numeric\_dtype(t[variabila]):  
 t[variabila].fillna(t[variabila].mean(), inplace=True)  
 else:  
 modulul = t[variabila].mode()[0]  
 t[variabila].fillna(modulul, inplace=True)  
  
  
def calcul\_procente(t, variabile, variabila\_raport):  
 p = t[variabile] \* 100 / t[variabila\_raport]  
 return p  
  
  
def vot\_majoritar(t):  
 k = np.argmax(t.iloc[1:]) + 1  
 return pd.Series(  
 data={"Localitate": t["City"],  
 "Partid majoritar": t.index[k]}  
 )

**Testul 5**

În fișierul ***Innoptari.csv*** se află informații privind numărul de înnoptări în structuri de primire turistică la nivel de localități conform INSSE. Informațiile sunt următoarele: Siruta - Codul Siruta al localității; Ani - Anul la care s-a făcut raportarea; Valoare - Număr de înnoptări. În fișierul CoduriRomania.xlsx sunt codificările localităților, județelor, regiunilor și macroregiunilor României conform NUTS (Nomenclatorul Unităților Teritoriale pentru Statistică din România). În fișierul ***PopulatieLocalitati.csv*** se află populația la nivelul localităților.

1**. Să se salveze în fișierul *Innoptari2016.csv* valorile privind înnoptările la nivelul anului 2016 la nivel de localitate. Se va salva pentru fiecare localitate codul Siruta, numele localității și numărul de înnoptări.**

**2. Să se salveze în fișierul *InnoptariJudete2015.csv* valorile privind înnoptările pe județe la nivelul anului 2015.**

**3. Să se salveze în fișierul *InnoptariJudeteAni.csv* numărul mediu de înnoptări pe județe și ani. Se va calcula media ponderată cu populația localităților.**

**4. Să se salveze în fișierul *InnoptariAni.csv* valorile privind înnoptările la nivel de localitate, pe ani. Valorile vor fi așezate pe coloane separate, câte o coloană pentru fiecare an.**

**5. Să se salveze în fișierul *InnoptariJudete.csv* valorile privind înnoptările la nivel de județ, pe ani. Valorile vor fi așezate pe coloane separate, câte o coloană pentru fiecare an.**

*Main.py*

import pandas as pd  
import functii as f  
  
innoptari = pd.read\_csv("innoptari.csv")  
f.nan\_replace(innoptari)  
# print(innoptari)  
localitati = pd.read\_excel("CoduriRomania.xlsx", index\_col=0)  
populatie = pd.read\_csv("PopulatieLocalitati.csv", index\_col=0)  
  
# Cerinta 1  
t1 = innoptari.merge(right=localitati,  
 left\_on="Siruta",  
 right\_index=True)  
t1\_ = t1[t1["Ani"] == 2016]  
# print(t1\_)  
t1\_[["Siruta", "City", "Valoare"]].to\_csv("Innoptari2016.csv", index=False)  
  
# Cerinta 2  
t2 = t1[t1["Ani"] == 2015][["County", "Valoare"]].groupby(by="County").agg(sum)  
t2.to\_csv("InnoptariJudete2015.csv")  
  
# Cerinta 3  
t1\_\_ = t1[["Siruta", "County", "Ani", "Valoare"]]. \  
 merge(right=populatie, left\_on="Siruta", right\_index=True)  
t3 = t1\_\_.groupby(by=["County", "Ani"]). \  
 apply(func=f.medie\_ponderata, pondere="Populatie", valoare="Valoare")  
t3.to\_csv("InnoptariJudeteAni.csv")  
  
# Cerinta 4  
t4 = innoptari.pivot(index="Siruta", columns="Ani", values="Valoare")  
# print(t4)  
t4.fillna(0, inplace=True)  
t4.to\_csv("InnoptariAni.csv")  
  
# Cerinta 5  
t5\_ = t4.merge(localitati, left\_index=True, right\_index=True)  
t5 = t5\_.groupby(by="County").agg(sum)  
t5.to\_csv("InnoptariJudete.csv")

*Functii.py*

import pandas as pd  
from pandas.api.types import is\_numeric\_dtype  
import numpy as np  
  
  
def nan\_replace(t):  
 assert isinstance(t, pd.DataFrame)  
 variabile = list(t.columns)  
 for variabila in variabile:  
 if any(t[variabila].isna()):  
 if is\_numeric\_dtype(t[variabila]):  
 t[variabila].fillna(t[variabila].mean(), inplace=True)  
 else:  
 modulul = t[variabila].mode()[0]  
 t[variabila].fillna(modulul, inplace=True)  
  
  
def medie\_ponderata(t, pondere, valoare):  
 assert isinstance(t, pd.DataFrame)  
 media = np.average(t[valoare], weights=t[pondere])  
 return media

**Seminarul 7 – recapitulare**

import pandas as pd  
import numpy as np  
  
print ("test")  
#eliminare randuri de inregistrari cu valori 0 pe linie  
#grupati in functie de tara etc  
df = pd.DataFrame({  
 'a': [0,0,1,1],  
 'b':[0,1,0,1]  
})  
  
print (df)  
  
df = df[(df.T!= 0).any()]  
  
#sau cu replace (0, NaN)  
  
#method to eliminate rows with NaN  
  
#calcul medie ponderata  
  
index = pd.Index(['01/01/2012','01/01/2012','01/01/2012','01/02/2012','01/02/2012'], name='Date')  
  
df = pd.DataFrame({'ID':[100,101,102,201,202],'ponderi':[.5,.75,1,.5,1],'valori':[60,80,100,100,80]},index=index)  
print (df)  
  
df2 = df.groupby(df.index).apply(lambda x:np.average(x.valori, weights = x.ponderi))  
print(df2)  
castig\_net = pd.read\_excel("CastigEuropa.xlsx")  
print(castig\_net)  
  
zone = pd.read\_excel("Zone.xlsx")  
print(zone)  
'''‘’’1) Să se determine pentru fiecare Tara media castigurilor salariale nete pe intreage perioada.   
Datele vor fi salvate in fisierul Ceminta1.csv, pe trei coloane: regiune, numele tarii si media castigurilor salariale nete. ‘’’ '''  
data = pd.merge(castig\_net, zone, how = "inner", on = "indicativ")  
print(data)  
ani = ['a2016','a2017','a2018', 'a2019']  
data['medie\_salariu\_tara'] = data[[col for col in ani]].mean(axis =1)  
print(data[['Tara', 'Regiune', 'medie\_salariu\_tara']])  
  
data[['Tara', 'Regiune', 'medie\_salariu\_tara']].to\_csv('Cerinta1.csv')