Python Exercise

Όνομα : Ιων Μπουρνάκας

ΑΜ: 1075475

Έτος: 3ο

Ερώτημα 1

* Να εξαχθεί γράφημα για τις συνολικές ποσότητες ανακυκλώσιμων ειδών ανά έτος

Αρχικά ο κώδικας για αυτό το κομμάτι είναι:

Text

Description automatically generated

Όπως φαίνεται έχουν χρησιμοποιηθεί βιβλιοθήκες όπως η panda που χρησιμοποιήθηκε για τον χειρισμό των δεδομένων στο csv αρχείο που είναι αποθηκευμένα τα data, η re (για regular expression), και η matplotlib.pyplot που χρησιμοποιήθηκε για τα plot . Μπορεί να μην φαίνεται στην φωτογραφία αλλά το data είναι ορισμένο με το αρχείο csv, που περιέχει όλα τα στοιχεία του πίνακα από το site , που έχει εξαχθεί πιο πάνω. Μετα φτιάχνουμε ένα list year\_to \_tons\_map το οποίο με την παραπάνω for πάμε και φορτώνουμε στην λίστα όλες τις γραμμές του πίνακα που περιέχουν στο πεδίο type το string "Recycl". Αυτό δηλαδή βρίσκει όλα τα ανακυκλώσιμα ειδή στον αρχικό πίνακα. Μετα αφού έχουμε εισάγει τις γραμμές πάμε και παίρνουμε από την κάθε γραμμή το date και κρατάμε από αυτό το string μόνο το έτος αφού μόνο αυτό μας ενδιαφέρει. Τέλος με την χρήση μιας if πάμε και ψάχνουμε για κάθε χρόνο όλα τα ανακυκλώσιμα και το άθροισμα όλων τα αναθέτουμε στο αντίστοιχο έτος. Μετα το plot με το plot τυπώνουμε για κάθε χρόνο τους συνολικούς τόνους ανακυκλώσιμων σε ένα διάγραμμα οπού το χ είναι τα έτη και το y οι τόνοι.

Αποτέλεσμα:

Chart, bar chart

Description automatically generated

Ερώτημα 2

* Συνολική παρουσίαση ανακυκλώσιμων ειδών και των αντίστοιχων

ποσοτήτων τους

Αρχικά ο κώδικας για αυτό το κομμάτι είναι:

Text

Description automatically generated

Φτιάχνουμε ένα list type\_to \_tons\_map το οποίο με την παραπάνω for πάμε και φορτώνουμε στην λίστα όλες τις γραμμές του πίνακα που περιέχουν στο πεδίο type το string "Recycl". Αυτό δηλαδή βρίσκει όλα τα ανακυκλώσιμα είδη στον αρχικό πίνακα. Μετα αφού έχουμε εισάγει τις γραμμές πάμε και παίρνουμε από την κάθε γραμμή τους τόνους του κάθε είδους. Τέλος με την χρήση μιας if πάμε και ψάχνουμε για κάθε ανακυκλώσιμο είδος τους τόνους και το άθροισμα όλων των τόνων του ιδίου είδους τα αναθέτουμε στο αντίστοιχο είδος . Μετα το plot με το plot τυπώνουμε για κάθε ανακυκλώσιμο είδος τους συνολικούς τόνους σε ένα διάγραμμα οπού το χ είναι το ανακυκλώσιμο είδος και το y οι τόνοι.

Αποτέλεσμα:

Chart, bar chart

Description automatically generated

Ερώτημα 3

* Παρουσίαση των 5 μηνών με τη μεγαλύτερη ποσότητα ανακυκλώσιμων

ειδών, ανεξαρτήτως έτους και είδους ανακυκλώσιμων ειδών

Αρχικά ο κώδικας για αυτό το κομμάτι είναι:

Text

Description automatically generated

Εδώ όπως φαίνεται τα πράγματα είναι λίγο πιο περιπλοκά. Έχουμε πάλι ένα list month\_to\_tons\_map και Μετα γεμίζουμε αυτό το list ακριβώς όπως γεμίσαμε το list στο πρώτο ερώτημα με την μονή διαφορά αντί από το stiring του date κρατάμε τον μηνά αντί για τον χρόνο. Μετα όμως όπως σε αυτή την λίστα έχουμε και τους 12 μήνες με το αντίστοιχο τους άθροισμα τόνων ανακυκλώσιμων και όχι τους 5 μεγαλύτερους όπως μας ζητείται. Αυτό γίνεται στο δεύτερο μισό του παραπάνω κώδικα οπού τώρα με την εντολή sorted βρίσκουμε τους 5 μήνες με τους περισσότερους τόνους στην λίστα μας. Τώρα με μια νέα list που ονομάζουμε top\_5\_months βάζουμε σαν κλειδιά της λίστας τους 5 ταξινομημένους μήνες που βρήκαμε από πάνω από την άλλη λίστα και για τιμή κλειδιών τις αντίστοιχες πάλι τιμές τους. Αρά τώρα έχουμε μια νέα λίστα με τους 5 μήνες με τους περισσότερους τόνους ανακυκλώσιμων. Και το διάγραμμα που προκύπτει από το plot με χ τους top 5 μήνες ταξινομημένους σε αύξουσα σειρά και το y οι συνολικοί τόνοι.

Το αποτέλεσμα :

Chart, bar chart

Description automatically generated

Οπού τα νούμερα είναι οι αντίστοιχοι μήνες του χρόνου στον χ

Τώρα όσον αναφορά την αποθήκευση αυτών των δεδομένων και αποτελεσμάτων σε μια βάση SQL. Όλη η διαδικασία μεταφοράς και μετασχηματισμού των δεδομένων έγινε στο παρών πρόγραμμα και σε γλώσσα python. Η βάση χρησιμοποιήθηκε μόνο για αποθήκευση.

Για τα δεδομένα του πρώτου ερωτήματος :

Ο κώδικας:

Text

Description automatically generated

Όπως βλέπουμε για το πρώτο ερώτημα βάλαμε και τις εντολές που κάνουν connect στην βάση που δημιουργήσαμε. Αυτή η βάση μιας και δημιουργήθηκε από εμένα δεν νομίζω να δουλεύει σε εσάς ο κώδικας χωρίς error. Για να δουλεύει θα πρέπει να φτιαχτεί μια ακριβώς ιδίων προδιαγραφών βάση από εσάς και να το τρέξετε (σε localhost με port 6033 username=keepo password=kappa και όνομα βάσης pythonproject).

Τώρα για τα δεδομένα με την χρήση εντολών που είδαμε στο μάθημα υπάρχει τρόπος να δίνουμε εντολές, που θα εκτελούσαμε μέσα στην βάση, για την δημιουργία και γέμισμα πινάκων με χρήση της python. Μια πολύ σημαντική είναι η cursor κλάση δημιουργεί αντικείμενα που μπορούν να εκτελέσουν λειτουργίες όπως εντολές SQL. Τα αντικείμενα αυτά αλληλοεπιδρούν με τον διακομιστή MySQL χρησιμοποιώντας ένα αντικείμενο MySQLConnection. Υστέρα με την εντολή create table year\_to\_tons με year και tons να είναι τύπου int φτιάχνουμε έναν άδειο πίνακα. Στον πίνακα αυτόν εισάγουμε τα δεδομένα από την λίστα που χρησιμοποιήθηκε στο πρώτο ερώτημα (year\_to\_tons\_map) που περιέχει όλα τα αποτελέσματα. Αυτή η εισαγωγή γίνεται με την insert που σαν όρισμα παίρνουν μια μια τις γραμμές τις λίστες και τις αποθηκεύουν στον πιάνα.

Με τον πίνακα τώρα να είναι έτσι:

Graphical user interface, text, application, email

Description automatically generated

Για τα δεδομένα του δευτέρου ερωτήματος :

Ο κώδικας:

Text, timeline

Description automatically generated

Με την cursor πάλι μπορούμε να καλέσουμε την create table type\_to\_tons οπού αυτή την φορά το type είναι είδους VARCHAR και το tons είδους INT. Τώρα τον πίνακα αυτόν τον γεμίζουμε με την εντολή insert και σαν δεδομένα παίρνει τα αποτελέσματα του δευτέρου ερωτήματος που είναι αποθηκευμένα στην λίστα type\_to\_tons\_map. Έτσι γραμμή γραμμή γίνεται εισαγωγή των αποτελεσμάτων του δευτέρου ερωτήματος στον νέο πίνακα της βασης που δημιουργήσαμε.

Με τον πίνακα τώρα να είναι έτσι:

Graphical user interface, text, application, email

Description automatically generated

Για τα δεδομένα του τρίτου ερωτήματος :

Ο κώδικας:

Text

Description automatically generated

Με την cursor πάλι μπορούμε να καλέσουμε την create table month\_to\_tons, με month και tons να είναι τύπου int φτιάχνουμε έναν άδειο πίνακα. Τώρα τον πίνακα αυτόν τον γεμίζουμε με την εντολή insert και σαν δεδομένα παίρνει τα αποτελέσματα του τρίτου ερωτήματος που είναι αποθηκευμένα στην λίστα top\_5\_months . Έτσι γραμμή γραμμή γίνεται εισαγωγή των αποτελεσμάτων του τρίτου ερωτήματος στον νέο πίνακα της βασης που δημιουργήσαμε.

Με τον πίνακα τώρα να είναι έτσι:

Graphical user interface, text, application, email

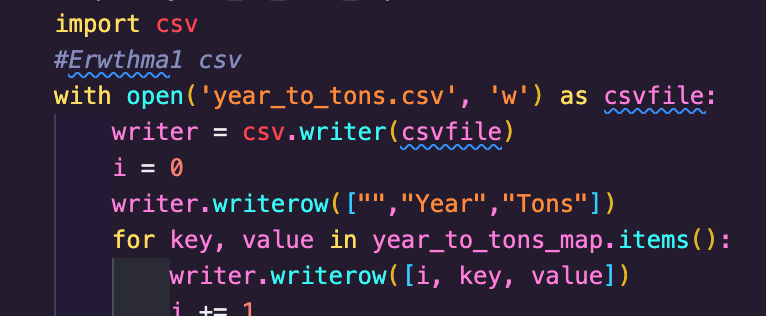
Description automatically generated

Τέλος το σχήμα της βασης είναι αυτό.

Graphical user interface, application

Description automatically generated

Τελικά για τα export των αποτελεσμάτων σε αρχεία csv χρησιμοποιήσαμε τον παρακάτω κώδικα:



Οπού αντίστοιχα για κάθε ερώτημα αλλάζει ποια λίστα εξάγουμε και το ονομα

Ολόκληρος ο κώδικας :

# %%

import sys

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from unicodedata import normalize

import requests

import re

from datetime import datetime

data=pd.read\_csv("data.csv")

# %%

#Erwthma 1

# find recycl in data['type']

year\_to\_tons\_map = {}

for i in range(len(data)):

if bool(re.search("Recycl[a-zA-Z]",data['type'][i])):

date = datetime.strptime(data['date'][i], '%Y-%m-%dT%H:%M:%S.%f').year

if date in year\_to\_tons\_map:

year\_to\_tons\_map[date] += data['total\_in\_tons'][i]

else:

year\_to\_tons\_map[date] = data['total\_in\_tons'][i]

#print(year\_to\_tons\_map[2022])

# print(year\_to\_tons\_map.keys)

# print(year\_to\_tons\_map.values)

years = list(year\_to\_tons\_map.keys())

tons = list(year\_to\_tons\_map.values())

plt.bar(range(len(year\_to\_tons\_map)), tons, tick\_label=years)

plt.xlabel("Years")

plt.ylabel("Number of tons")

plt.title("Number of tons of recyclables in each year")

plt.show()

# %%

#Erwthma 2

type\_to\_tons\_map = {}

for i in range(len(data)):

if bool(re.search("Recycl[a-zA-Z]",data['type'][i])):

type = data['type'][i]

if type in type\_to\_tons\_map:

type\_to\_tons\_map[type] += data['total\_in\_tons'][i]

else:

type\_to\_tons\_map[type] = data['total\_in\_tons'][i]

# print(type\_to\_tons\_map)

# print(type\_to\_tons\_map.keys)

# print(type\_to\_tons\_map.values)

type = list(type\_to\_tons\_map.keys())

tons\_1 = list(type\_to\_tons\_map.values())

plt.bar(range(len(type\_to\_tons\_map)), tons\_1, tick\_label=type)

plt.xlabel("Types")

plt.ylabel("Number of tons")

plt.title("Number of tons of recyclables in each type")

plt.show()

# %%

#Erwthma 3

month\_to\_tons\_map = {}

for i in range(len(data)):

if bool(re.search("Recycl[a-zA-Z]",data['type'][i])):

date = datetime.strptime(data['date'][i], '%Y-%m-%dT%H:%M:%S.%f').month

if date in month\_to\_tons\_map:

month\_to\_tons\_map[date] += data['total\_in\_tons'][i]

else:

month\_to\_tons\_map[date] = data['total\_in\_tons'][i]

#print(year\_to\_tons\_map[2022])

# print(month\_to\_tons\_map.keys)

# print(month\_to\_tons\_map.values)

d = sorted(month\_to\_tons\_map, key=month\_to\_tons\_map.get)[::-1][0:5]

top\_5\_months = {}

for i in d:

top\_5\_months[i] = month\_to\_tons\_map[i]

top = list(top\_5\_months.keys())

tons\_2 = list(top\_5\_months.values())

plt.bar(range(len(top\_5\_months)), tons\_2, tick\_label=top)

plt.xlabel("Top 5 Months")

plt.ylabel("Number of tons")

plt.title("Top 5 months with the most recyclables")

plt.show()

# %%

#SQL1

import mysql.connector

mydb = mysql.connector.connect(

host="localhost",

user="keepo",

passwd="kappa",

database="pythonproject",

port=6033

)

mycursor = mydb.cursor()

# drop table if exists

mycursor.execute("DROP TABLE IF EXISTS year\_to\_tons")

mycursor.execute('CREATE TABLE year\_to\_tons (year INT NOT NULL , tons INT NOT NULL )')

for i in range(len(year\_to\_tons\_map)):

data = (years[i], int(tons[i]))

mycursor.execute("INSERT INTO year\_to\_tons (year, tons) VALUES (%s, %s)", (data))

mydb.commit()

# %%

#SQL2

mycursor\_1 = mydb.cursor()

# drop table if exists

mycursor\_1.execute("DROP TABLE IF EXISTS type\_to\_tons")

mycursor\_1.execute('CREATE TABLE `type\_to\_tons` (`type` VARCHAR(100) NOT NULL , `tons` INT NOT NULL )')

for i in range(len(type\_to\_tons\_map)):

data = (type[i], int(tons\_1[i]))

mycursor\_1.execute("INSERT INTO type\_to\_tons (type, tons) VALUES (%s, %s)", (data))

mydb.commit()

# %%

#SQL3

mycursor\_2 = mydb.cursor()

# drop table if exists

mycursor\_2.execute("DROP TABLE IF EXISTS month\_to\_tons")

mycursor\_2.execute('CREATE TABLE `month\_to\_tons` (`month` INT NOT NULL , `tons` INT NOT NULL )')

for i in range(len(top\_5\_months)):

data = (top[i], int(tons\_2[i]))

mycursor\_2.execute("INSERT INTO month\_to\_tons (month, tons) VALUES (%s, %s)", (data))

mydb.commit()

# %%

#export dictionary to csv

#export year\_to\_tons\_map to csv

import csv

#Erwthma1 csv

with open('year\_to\_tons.csv', 'w') as csvfile:

writer = csv.writer(csvfile)

i = 0

writer.writerow(["","Year","Tons"])

for key, value in year\_to\_tons\_map.items():

writer.writerow([i, key, value])

i += 1

# %%

#Erwthma2 csv

with open('type\_to\_tons.csv', 'w') as csvfile:

writer = csv.writer(csvfile)

i = 0

writer.writerow(["","Type","Tons"])

for key, value in type\_to\_tons\_map.items():

writer.writerow([i, key, value])

i += 1

# %%

#Erwthma3 csv

with open('top\_5\_months.csv', 'w') as csvfile:

writer = csv.writer(csvfile)

i = 0

writer.writerow(["","Month","Tons"])

for key, value in top\_5\_months.items():

writer.writerow([i, key, value])

i += 1