

Προαιρετική άσκηση 2019-2020

Python

Κώδικας python

Αρχικά να διευκρινιστεί ότι το πρόγραμμα αποτελείται από τέσσερα αρχεία κώδικα τα οποία με την σειρά τους περιέχουν συναρτήσεις οι οποίες πρέπει να καλεστούν manually ώστε να τρέξουν τα scripts που βρίσκονται μέσα σε αυτές. Τα αρχεία αυτά είναι: `get_data.py`, `download.py`, `connection.py`, `plots.py`. Επιπλέον υπάρχουν τρεις ακόμα φάκελοι που περιέχουν τα αρχεία html στα οποία έγινε το web scrapping για να κατέβουν τα αρχεία excel (φάκελος excel και αρχείο `download.py`) και τα csv στα οποία έγιναν μικρές αλλαγές σε κάποιες σειρές των αρχικών excel. Τέλος κάποια προβλήματα που παρουσιάστηκαν, λόγω της ανομοιομορφίας στα ονόματα, κατά την εξαγωγή δεδομένων από τα excel, αντιμετωπίστηκαν χρησιμοποιώντας reference by index και όχι by value ή name, στο Pandas.

Αρχείο `download.py`

Το αρχείο `download.py` έχει ως λειτουργία το parsing της σελίδας της ΕΛΣΤΑΤ ώστε να βρεθούν τα excel που θα χρησιμοποιηθούν για την εξαγωγή των δεδομένων.

Αρχικά καλείται η συνάρτηση `download_html` η οποία χρησιμοποιώντας το module `os`, ελέγχει (αν υπάρχει) αν υπάρχουν οι φάκελοι και τους δημιουργεί (στο `dir` του πρότζεκτ). Σε αυτούς θα αποθηκευτούν τόσο τα αρχεία excel όσο και τα html. Παρατηρούμε ότι οι διευθύνσεις των σελίδων της ΕΛΣΤΑΤ που θέλουμε να κατεβάσουμε είναι της μορφής «<https://www.statistics.gr/el/statistics/-/publication/STO04/> + έτος + τρίμηνο». Με την χρήση της `requests` κατεβάζουμε τα html αρχεία και τα αποθηκεύουμε, χρησιμοποιώντας βρόχους επανάληψης ανάμεσα στα έτη και τα τρίμηνα ώστε να προσδιοριστούν οι μεταβλητές «έτος» και «τρίμηνο».

Στη συνέχεια καλείται η δεύτερη συνάρτηση `download_excel` η οποία κάνει parsing στα html που κατέβασε η προηγούμενη συνάρτηση, για να κατεβάσει τα excel. Το module που πραγματοποιεί αυτή τη λειτουργία είναι το `beautiful soup`.

Τέλος πραγματοποιούνται κάποιες ενέργειες «καλλωπισμού» στα αρχεία excel όπως στην διαγραφή κάποιων «άχρηστων» γραμμών που δυσχεραίνουν την ανάγνωση τους από το pandas.

```

import requests
import os from bs4
import BeautifulSoup
import pandas as pd

def download_html():
#download html files
for year in range(2011
,2015):
    if not os.path.exists("./html/" + str(year) + "/"):
        os.makedirs(os.path.dirname( "./html/" + str(year) +
"/"))
        for trimester in range(1 , 5):
            #Create URLs to download the html
            url = str(year) + "-Q" +
str(trimester)
            print("Downloading
HTML: " + url)

            #Download the html pages
html = 'https://www.statistics.gr/en/statistics/-
/publication/ST004/' + url
            r = requests.get(html, allow_redirects=True)
            #save it in its coresponding folder
open('./html/' + str(year) + '/' + str(trimester) + '.html', 'wb
').write(r.content)

```

```

def download_excel():
#Get Excel files for year
in range(2011 ,2015):
for trimester in range(1 ,
5):
    if not os.path.exists("./excel/" + str(year) + "/" +
str(trimester) +
"/" ):
        #make folders if they do not exist
os.makedirs(os.path.dirname( "./excel/" + str(year) + "/" + str(t
rimester) + "/" ))

        #Parse html and get URLs for excel files
soup =
BeautifulSoup(open("./html/" + str(year) + "/" + str(trimester
) + ".html", encoding='utf8'), 'html.parser')
href =
soup.findAll("table", {"class": "documentsTable"})[2].find_all
("a")[2]["href"]
print("Downloading excel file for year :" +
str(year) + " and trimester: " + str(trimester))

        #Download the excel files
excel = href
r = requests.get(excel,
allow_redirects=True)
        #save them in their corresponding folder
open('./excel/' + str(year) + '/' + str(trimester) + "/" + '1.xls',
'wb').write(r.content)
download_html()
download_excel()
for year
in range(2011 ,2013):
for trimester in range(1
, 5):
    df =
pd.read_excel("./excel/"+str(year)+"/"+str(trimester)+"/1.xls")
df.columns = ['index','countries','air','railway','sea','road','total']
df.drop(df.index[[0,1]])

for year in range(2013 ,2015):
    for trimester in range(1 , 5):
        df =
pd.read_excel("./excel/"+str(year)+"/"+str(trimester)+"/1.xls")
df.columns = ['index','countries','air','railway','sea','road','total']
df.drop(1)

```

Αρχείο get_data.py

Το αρχείο get_data.py έχει ως στόχο την εξαγωγή των ζητούμενων δεδομένων για την δημιουργία των γραφημάτων. Αποτελείται από 4 συναρτήσεις (όσα και τα γραφήματα που ζητούνται), την get_total_arrivals, την get_most_visits_per_country, την arrivals_by_means_of_transport , και την get_tourists_per_trimester.

Οι παραπάνω συναρτήσεις χρησιμοποιούν ως κύρια μέθοδο τις λούπες, είτε στο τελευταίο αρχείο excel κάθε χρονιάς, είτε το τελευταίο αρχείο excel κάθε εξαμήνου, αφού σε αυτά περιγράφεται όλη η απαραίτητη πληροφορία. Στις λούπες αυτές διαβάζεται κάθε αρχείο από κάτω προς τα πάνω ψάχνοντας τα δεδομένα που χρειάζεται κάθε γράφημα. Επίσης κάθε φορά που ένα excel εισάγεται pandas, γίνεται και μια ονοματοδοσία στις στήλες για την διευκόλυνση της αναζήτησης ανά στήλη.

Τέλος να αναφερθεί ότι η (προσωρινή) αποθήκευση των δεδομένων γίνεται dictionaries ή σε λίστες που αποτελούνται από dictionaries ή λίστες (list of dictionaries, list of lists). Επιπλέον για την εισαγωγή αυτών των δεδομένων στην MySQL και στην matplotlib, πολλές οι λίστες έγιναν flat (όπου flat lists <https://stackoverflow.com/questions/952914/how-to-make-a-flat-list-out-of-list-of-lists>) .

```
import pandas as pd from
collections import Counter
import os
def
get_total_arrivals():
    total_arrivals=0
arrivals=[]      no=-2
    #searching in the 4th excel file of every year and getting the last line
which h indicates total number of passengers      for year in range (2011,2015):
        df = pd.read_excel("./excel/"+str(year)+"/"+str(4)+"/1.xls", 'DEC')
df.columns = ['index', 'countries', 'air', 'railway', 'sea', 'road', 'total']
#this while loop is used because the last line of the excel doesnt really
indicates total number of passengers beacause last lines contain some text
while no > -5:          try:
            total_arrivals += int(df["total"].iloc[no])
arrivals.append(int(df["total"].iloc[no]))          break
```

```

        except:
no-=1
arrivals.append(total_arrivals)
return arrivals
def
get_most_visits_per_country():
    countries=[]    cntr={}
for year in range (2011,2015):
    i=-2    df =
pd.read_excel("./excel/"+str(year)+"/"+str(4)+"/1.xls",'DEC')
df.columns = ['index','countries','air','railway','sea','road','total']
df = df[df.countries != 'TOTAL ARRIVALS']    df = df[df.countries != 'of
which:']    while i>-62 :    if(not
pd.isnull(df['countries'].iloc[i])):
cntr[df.iloc[i,1]] = int(df["total"].iloc[i])

i-=1

    #adding to list of dictionaries named countries, the temp dictionary
cntr  which contains every year's total tourists per country
countries.append(cntr.copy())
    #export to csv
outname = str(year)+'.csv'
outdir = './csv'    if not
os.path.exists(outdir):
    os.mkdir(outdir)
    fullname = os.path.join(outdir,
outname)    df.to_csv(fullname)

    #suming every year from countries list of
dictionaries    total_countries = {}    for d in
countries:    for k in d.keys():
        total_countries[k] = total_countries.get(k,0) + d[k]
        sort_total = sorted(total_countries.items(), key=lambda x: x[1],
reverse=True
)
    return sort_total
get_most_visits_per_country() def
arrivals_by_means_of_transport():

```

```

#dictionary means will contain every mean of transport as key and their
total tourists as value      means={}      no=-2

#searching in the 4th excel file of every year and getting the last line
whic h indicates total number of passengers      for year in range (2011,2015):
    df = pd.read_excel("./excel/"+str(year)+"/"+str(4)+"/1.xls", 'DEC')
df.columns = ['index', 'countries', 'air', 'railway', 'sea', 'road', 'total']
    #this while loop is used because the last line of the excel doesnt
really indicates total number of passengers beacause last lines contain some
text      while no > -5:
        try:
            x=means.setdefault ("air" , int(df['air'].iloc[no]))
x=means.setdefault ("railway", int(df['railway'].iloc[no]))
x=means.setdefault ("sea", int(df['sea'].iloc[no]))
x=means.setdefault ("road", int(df['road'].iloc[no]))      break
        except:
            no-=1
    return
means
def
get_tourists_per_trimester():
    total=[]
    tempList=[]
    no=-2
    temp=0
    #searching in the 4th excel file of every year and getting the last line
whic h indicates total number of passengers      for year in range (2011,2015):

        xls = pd.ExcelFile("./excel/"+str(year)+"/"+str(4)+"/1.xls")
        #parse excel sheet every three months
for i in range (2,12,3):
    df=xls.parse(i)
    df.columns = ['index', 'countries', 'air', 'railway', 'sea', 'road', 'total
']
        #this while loop is used because the last line of the excel doesnt
re ally indicates total number of passengers beacause last lines contain some
text      while no > -5:
        try:
            #initializing my tempory variable and list every first
semest er of the year

```

```

        if((year==2012 or year==2013 or year == 2014) and i==2):
            temp=0
tempList.clear()
temp=sum(tempList)

        #insert in the end of our list, trimester's total
tourists
tempList.insert(len(tempList),int(df["total"].iloc[no])temp)
break          except:          no-=1
        #copying the temporary list that contains every year's tourists per
trime ster, to a list that contains all years
total.append(tempList.copy())
        #total is a list of lists so flat_total is total list
flattened      flat_total = [item for sublist in total for item in
sublist]      return (flat_total) get_tourists_per_trimester()

```

Αρχείο connection.py Σημείωση: προκειμένου να τρέξουν όλα τα αρχεία πρέπει να γραφτεί η εντολή
mycursor.execute("CREATE DATABASE elstat")

Στο αρχείο αυτό μεταφέρονται, τα δεδομένα από το προηγούμενο αρχείο (get_data.py) στην βάση δεδομένων. Αποτελείται από ισάριθμες συναρτήσεις με το get_data.py αφού κάθε μια συνάρτηση έχει ως στόχο την μεταφορά και δημιουργία table για κάθε ζητούμενο γράφημα. Σε κάποιες από τις συναρτήσεις όπως η tourism_per_semester εισάγονται παραπάνω δεδομένα για την καλύτερη απεικόνιση τους στην βάση αλλά και στα διαγράμματα. Στην συγκεκριμένη για παράδειγμα εισάγονται και όλα τα τρίμηνα με την μορφή <έτος> + <τρίμηνο> όπου έτος από το 2011 μέχρι το 2015 και τρίμηνο από τις τιμές a, b, c, d.

```

import pandas as pd
import mysql.connector
from get_data import get_most_visits_per_country, get_total_arrivals, arrivals_by_means_of_transport, get_tourists_per_trimester
#all the data that are being uploaded below come from get_data.py's functions that are imported above
mydb =
mysql.connector.connect(
host="localhost",
user="root",
password="password",
database="elstat"
)
#uploading total arrivals to database
def total_arrivals_mysql():
    mycursor = mydb.cursor()
    mycursor.execute("CREATE TABLE IF NOT EXISTS total_arrivals (year VARCHAR(255), arrivals INT (10));")
    years=['2011', '2012', '2013', '2014', 'total']
    sql = "INSERT INTO total_arrivals (year, arrivals) VALUES (%s, %s);"
    for elem in zip(years, get_total_arrivals()):
        mycursor.execute(sql, elem)
    mydb.commit()

#uploading most visited tourists per country
def countries_to_mysql():
    mycursor = mydb.cursor()
    mycursor.execute("CREATE TABLE IF NOT EXISTS countries (country VARCHAR(255), no INT (10));")
    sql = "INSERT INTO countries (country, no) VALUES (%s, %s);"
    mycursor.executemany(sql, get_most_visits_per_country())
    mydb.commit()
    mycursor.execute("SELECT * FROM countries;")
    for x in mycursor:
        print (x)

#means of transport statistics upload to mysql server
def mot_to_mysql():
    mycursor = mydb.cursor()

```



```

mycursor.execute("CREATE TABLE IF NOT EXISTS means_of_transport (means VARCHAR(
255), no INT (10));") newdict=arrivals_by_means_of_transport()
means=list(newdict.keys()) number=list(newdict.values())
sql='INSERT INTO `means_of_transport` (`means`,`no`) values (%s,%s);'
for elem in zip(means, number): mycursor.execute(sql, elem)
mydb.commit()

mot_to_mysql()
#uploading tourists per trimester
def tourism_per_trimester(): mycursor =
mydb.cursor() mycursor.execute("CREATE TABLE IF NOT EXISTS trimesters
(trimester VARCHAR(255)
, no INT(12) );")
varlist=get_tourists_per_trimester() dates=[
("2011a"),("2011b"),("2011c"),("2011d"),

("2012a"),("2012b"),("2012c"),("2012d"),("2013a"),("201 3b"),("2013c"),("2013d"),
("2014a"),("2014b"),("2014c"),("2014d")]
sql='INSERT INTO `trimesters` (`trimester`,`no`) values (%s,%s);'
for elem in zip(dates, varlist): mycursor.execute(sql, elem)
mydb.commit()

```

Αρχείο plots.py

Σε αυτό το αρχείο γίνεται εμφάνιση των δεδομένων που έχουν αποθηκευτεί στη βάση μέσω της matplotlib. Αποτελείται ξανά από 4 συναρτήσεις όσες και τα ζητούμενα γραφήματα.

Αρχικά συνδέεται στην βάση και από εκεί ανακτά τα αποθηκευμένα δεδομένα τοποθετώντας τα σε λίστες. Αν τα δεδομένα δεν έχουν γίνει flat από κάποιο προηγούμενο ερώτημα, γίνονται. Στην συνέχεια χρησιμοποιούνται συναρτήσεις της matplotlib, όπως figure για καθοριστεί το μέγεθος, οι x και y label για να δοθούν ονόματα στις γραμμές και τις στήλες αντίστοιχα, η yticks για να γίνει αλλαγή στο scale του y άξονα και η bar ώστε να εμφανιστούν τα δεδομένα στο γράφημα με την μορφή κάθετης μπάρας. Τέλος γίνεται εμφάνιση των παραπάνω με την συνάρτηση show.

Ο λόγος που χρησιμοποιήθηκε η βάση ως διαμεσολαβητής μεταξύ του αρχείου get_data.py και plot.py είναι για να γίνουν πιο «όμορφα» τα δεδομένα που εμφανίζονται αλλά και να δοθεί κάποια αξία στην

βάση ώστε να μην «υπάρχει» απλά. Με αυτή την έννοια ακολουθείται μια διαδικασία που εν συντομία περιλαμβάνει: 1) το κατέβασμα των αρχείων από την ΕΛΣΤΑΤ, 2) την επεξεργασία και αποθήκευση τους τοπικά με το Pandas, 3) το ανέβασμα τους σε μια βάση δεδομένων και τέλος 4) το κατέβασμα από την βάση και εμφάνιση τους με την matplotlib .

```
import matplotlib.pyplot as
plt import numpy as np import
mysql.connector
mydb =
mysql.connector.connect(
host="localhost",
user="root",
password="Bilbo83os",
database="elstat"
) def
plot_trimesters():
```

```

    #download data from database    mycursor =
mydb.cursor()    mycursor.execute("select trimester
from trimesters")    column = mycursor.fetchall()
mycursor.execute("select no from trimesters")    row =
mycursor.fetchall()

    #making list of lists flat to use them in plots    flat_row =
[item for sublist in row for item in sublist]    flat_column =
[item for sublist in column for item in sublist]
    #ploting as bars    plt.figure(figsize=(12, 9))
plt.yticks(np.arange(0, max(flat_row), 1000000))
plt.xlabel("trimsters")    plt.ylabel("tourists
per trimester x10^7")
plt.bar(flat_column,flat_row,align='center')

plt.show()
def
plot_countries():
    #download data from database    mycursor =
mydb.cursor()    mycursor.execute("select country
from countries")    column = mycursor.fetchall()
mycursor.execute("select no from countries")    row
= mycursor.fetchall()

    #making list of lists flat to use them in plots    flat_row =
[item for sublist in row for item in sublist]    flat_column =
[item for sublist in column for item in sublist]
    #ploting as bars
    plt.figure(figsize=(16, 9))
    plt.yticks(np.arange(0, max(flat_row), 1000000))
plt.xticks(rotation='vertical')
plt.xlabel("countries")
    plt.ylabel("tourists per country")
    plt.bar(flat_column,flat_row,align='center')

    plt.show()
def
plot_means_of_transport():
#download data from database
mycursor = mydb.cursor()

```

```

    mycursor.execute("select means from
means_of_transport")    column = mycursor.fetchall()
mycursor.execute("select no from means_of_transport")
row = mycursor.fetchall()

    #making list of lists flat to use them in plots    flat_row =
[item for sublist in row for item in sublist]    flat_column =
[item for sublist in column for item in sublist]
    #ploting as bars    plt.figure(figsize=(5, 5))
plt.yticks(np.arange(0, max(flat_row), 1000000))
    plt.xlabel("means of transport")
plt.ylabel("numbers of means used")
    plt.bar(flat_column,flat_row,align='center')

plt.show()

def plot_total_arrivals():    #download data from
database    mycursor = mydb.cursor()
mycursor.execute("select year from total_arrivals")
column = mycursor.fetchall()    mycursor.execute("select
arrivals from total_arrivals")    row =
mycursor.fetchall()

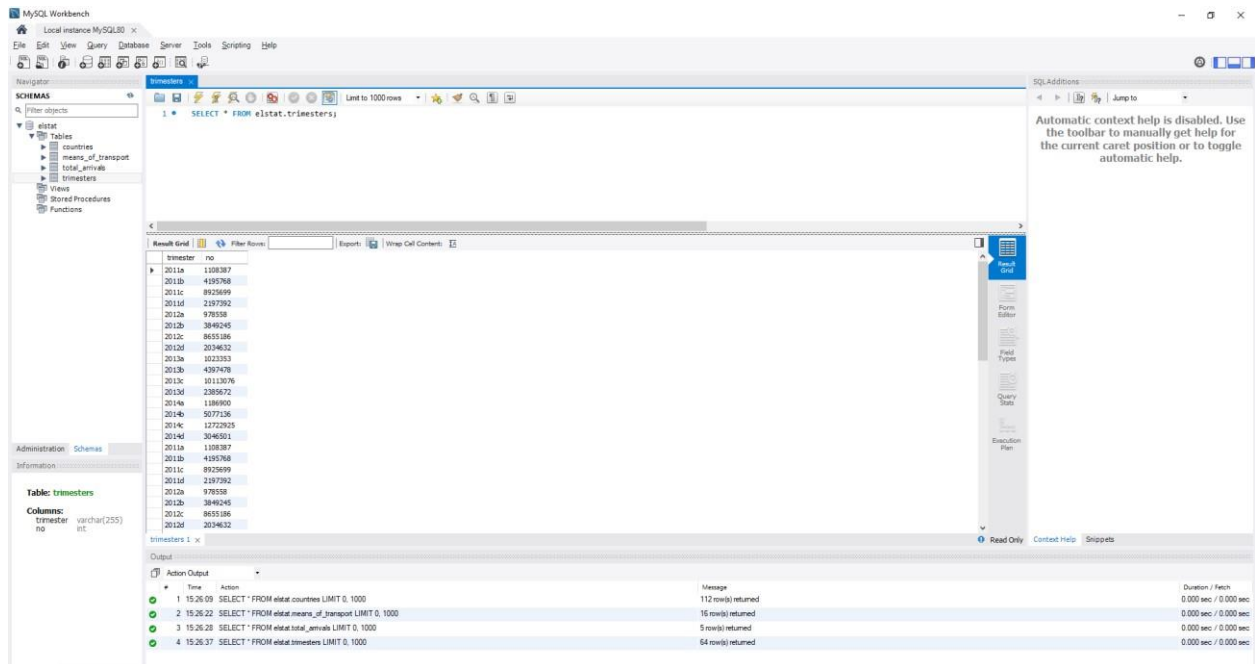
    #making list of lists flat to use them in plots    flat_row =
[item for sublist in row for item in sublist]    flat_column =
[item for sublist in column for item in sublist]
    #ploting as bars
plt.figure(figsize=(5, 5))
    plt.yticks(np.arange(0, max(flat_row),
100000000))    plt.xlabel("year")
plt.ylabel("arrivals")
    plt.bar(flat_column,flat_row,align='center')

plt.show()

```

Παραδείγματα εφαρμογής

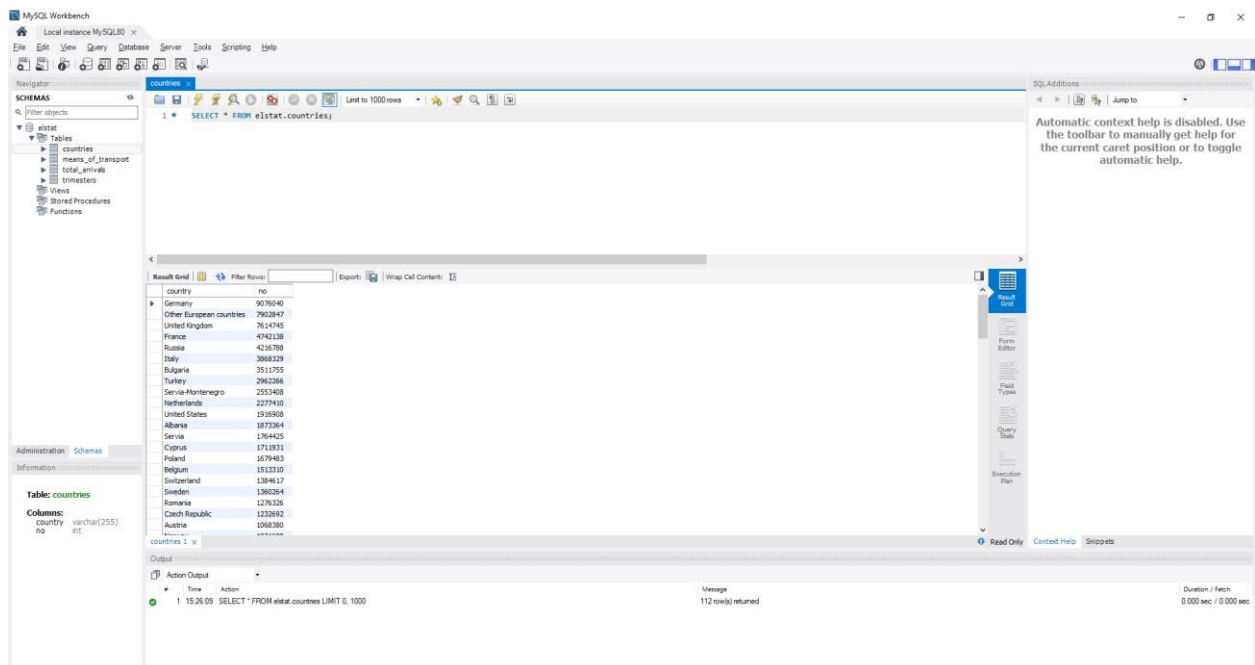
Η βάση δεδομένων με όλα τα στοιχεία που περιέχει



MySQL Workbench interface showing the 'elstat' database schema. The 'trimesters' table is selected, displaying columns: 'country' (varchar(255)) and 'no' (int). The output shows a list of countries and their corresponding 'no' values.

country	no
Germany	9076040
Other European countries	7602847
United Kingdom	7614745
France	4742138
Russia	4218780
Italy	3868329
Bulgaria	3511755
Turkey	2962266
Serbia-Montenegro	2557408
Netherlands	2277410
United States	1910908
Albania	1873364
Serbia	1764425
Cyprus	1711831
Poland	1679483
Belgium	1513310
Switzerland	1384617
Sweden	1360284
Romania	1276326
Czech Republic	1232892
Austria	1068380

Εικόνα 1 όλα τα τρίμηνα με τους τουρίστες τους



MySQL Workbench interface showing the 'elstat' database schema. The 'countries' table is selected, displaying columns: 'country' (varchar(255)) and 'no' (int). The output shows a list of countries and their corresponding 'no' values.

country	no
Germany	9076040
Other European countries	7602847
United Kingdom	7614745
France	4742138
Russia	4218780
Italy	3868329
Bulgaria	3511755
Turkey	2962266
Serbia-Montenegro	2557408
Netherlands	2277410
United States	1910908
Albania	1873364
Serbia	1764425
Cyprus	1711831
Poland	1679483
Belgium	1513310
Switzerland	1384617
Sweden	1360284
Romania	1276326
Czech Republic	1232892
Austria	1068380

Εικόνα 2 σύνολο χωρών ταξινομημένο σύμφωνα με τους περισσότερους τουρίστες

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'elstat' database schema with tables 'countries', 'means_of_transport', 'total_arrivals', and 'trimesters'. The main query editor contains the SQL statement: `SELECT * FROM elstat.means_of_transport;`. The 'Results' tab shows the following data:

means	no
air	11671154
railway	3765
sea	947847
road	3004479

The bottom 'Output' tab shows the execution log:

#	Time	Action	Message	Duration / Fetch
1	15:26:09	SELECT * FROM elstat.countries LIMIT 0, 1000	112 row(s) returned	0.000 sec / 0.000 sec
2	15:26:22	SELECT * FROM elstat.means_of_transport LIMIT 0, 1000	16 row(s) returned	0.000 sec / 0.000 sec

Εικόνα 3 τα μέσα που χρησιμοποίησαν οι τουρίστες για να επισκεφτούν την Ελλάδα

The screenshot shows the MySQL Workbench interface. The left sidebar displays the 'elstat' database schema. The main query editor contains the SQL statement: `SELECT * FROM elstat.total_arrivals;`. The 'Results' tab shows the following data:

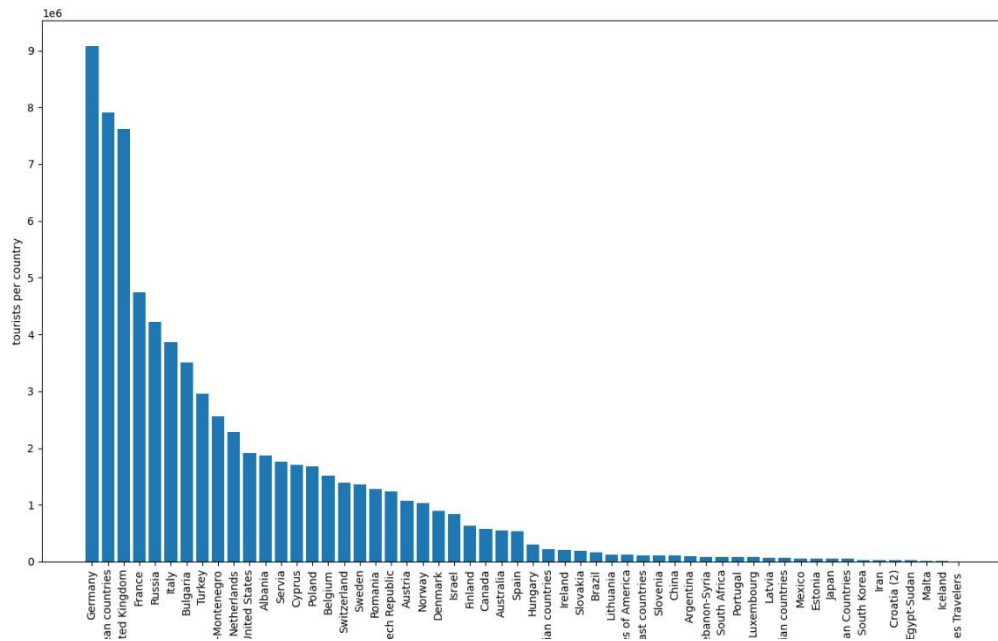
year	arrivals
2011	16427246
2012	18517621
2013	17919579
2014	22033462
total	71897908

The bottom 'Output' tab shows the execution log:

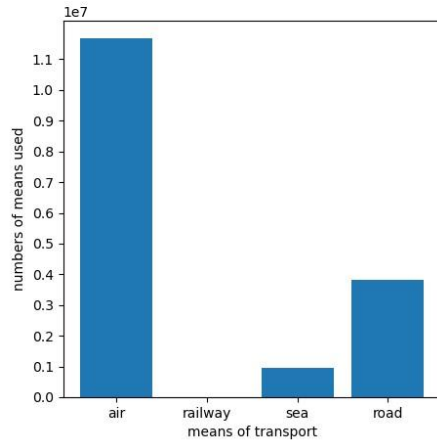
#	Time	Action	Message	Duration / Fetch
1	15:26:09	SELECT * FROM elstat.countries LIMIT 0, 1000	112 row(s) returned	0.000 sec / 0.000 sec
2	15:26:22	SELECT * FROM elstat.means_of_transport LIMIT 0, 1000	16 row(s) returned	0.000 sec / 0.000 sec
3	15:26:28	SELECT * FROM elstat.total_arrivals LIMIT 0, 1000	5 row(s) returned	0.000 sec / 0.000 sec

Εικόνα 4 τουρίστες αν χρονιά και συνολικά

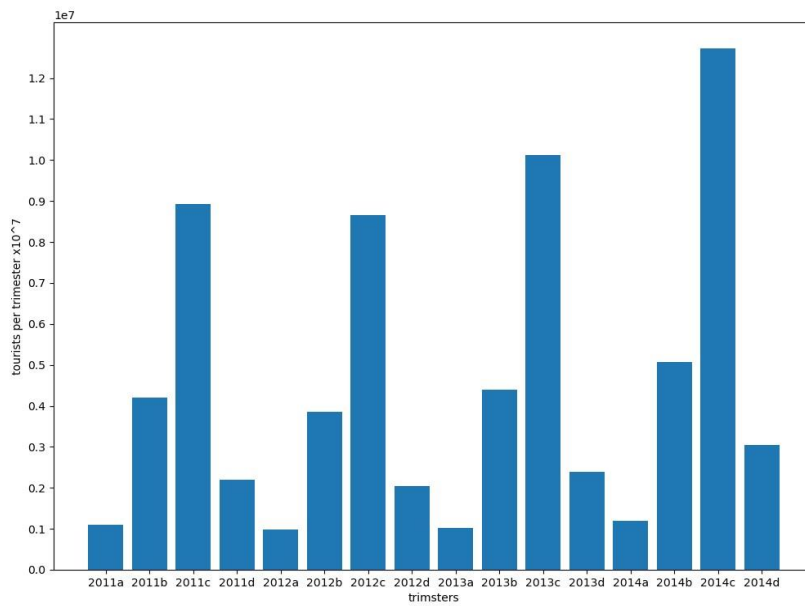
Γραφήματα



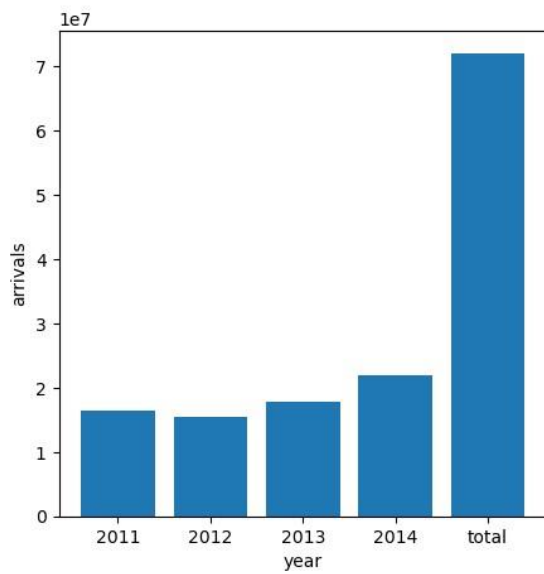
Εικόνα 5 τουρίστες ανά χώρα



Εικόνα 6 τουρίστες ανά μέσο



Εικόνα 7 τουρίστες ανά εξάμηνο



Εικόνα 8 τουρίστες ανά χρονιά