

# ΠΡΟΑΙΡΕΤΙΚΗ ΕΡΓΑΣΤΗΡΙΑΚΗ ΑΣΚΗΣΗ

## PYTHON

Πάτρα 2023-2024

**Όνοματεπώνυμο:** Γεώργιος Μερμίγκης

**AM:** 1084639

**Έτος:** 3ο



## **Περιεχόμενα:**

- A. Παραδοχές
- B. Επεξήγηση κώδικα με screenshots και παραδείγματα
- C. SQL, CSV κώδικας
- D. Graphical User Interface (GUI)
- E. Βιβλιογραφία
- F. Συνολικός τελικός κώδικας

## **Α) Παραδοχές:**

- Στο πρώτο ζητούμενο που αφορά την παρουσίαση τζίρου ανά μήνα, επέλεξα να αξιοποιήσω το συνολικό τζίρο κάθε μήνα από το 2015-2021, αντί να δείξω τον τζίρο κάθε μήνα ξεχωριστά από τον 2015-2021 για το κάθε έτος.
- Στο δεύτερο ζητούμενο που αφορά την παρουσίαση τζίρου ανά χώρα, επέλεξα να μην εντάξω στο γράφημα μου τις περιπτώσεις όπου: Country="All" ή Country="Total (excluding China)", διότι οι 2 αυτές περιπτώσεις έχουν μεγάλη απόκλιση από τις υπόλοιπες.
- Στο τρίτο ζητούμενο που αφορά την παρουσίαση τζίρου ανά μέσο μεταφοράς, επέλεξα να μην εντάξω στο γράφημα μου την περίπτωση όπου: Transport\_Mode="All". Θεώρησα, πως αφού έχουμε τις άλλες 2 περιπτώσεις (Sea, Air), η περίπτωση "All" είναι περιττή και έχει μεγάλη απόκλιση από τις άλλες 2 περιπτώσεις.
- Στο τέταρτο ζητούμενο που αφορά την παρουσίαση τζίρου ανά ημέρα της εβδομάδας, χρησιμοποίησα τη στήλη "Date" του .csv αρχείου που μας δόθηκε και όχι τη στήλη "Weekday".
- Στο πέμπτο ζητούμενο που αφορά την παρουσίαση τζίρου ανά κατηγορία εμπορεύματος, επέλεξα να μην εντάξω στο γράφημα μου την περίπτωση όπου: Commodity="All", γιατί εφόσον περιλαμβάνω τις υπόλοιπες περιπτώσεις του Commodity, πιστεύω δεν είναι απαραίτητο.
- Στο έκτο ζητούμενο που αφορά την παρουσίαση των 5 μηνών με το μεγαλύτερο τζίρο, επέλεξα να δείξω τους 5 μήνες με το μεγαλύτερο συνολικό τζίρο, από τα 6 έτη που αφορά το .csv αρχείο (δεύτερη σκέψη μου ήταν να βγάλω τους 5 unique πιο "δυνατούς" μήνες των 6 ετών).
- Στο έβδομο ζητούμενο που αφορά την παρουσίαση των 5 κατηγοριών εμπορευμάτων με το μεγαλύτερο τζίρο για κάθε χώρα, δεν έχω συμπεριλάβει την περίπτωση όπου: Commodity="All".

## **B) Επεξήγηση κώδικα με screenshots και παραδείγματα:**

Αρχικά, πρέπει να κατεβάσω το αρχείο ώστε να συγκεντρώσω τα δεδομένα που θέλω να επεξεργαστώ. Δημιουργώ μια μεταβλητή url όπου και αναθέτω το link του .csv αρχείου.

Ενσωματώνω στον κώδικα μου την βιβλιοθήκη “requests”, “csv”, ώστε να μπορέσω να κατεβάσω και να χειριστώ το αρχείο μου.

Με το with δημιουργώ ένα session που επιτρέπει στον κώδικα να διατηρεί cookies σε όλες τις αιτήσεις. Καλώ την get μέθοδο με παράμετρο το url, που όρισα παραπάνω, ώστε να ανακτήσω το περιεχόμενο του .csv αρχείου από το καθαρισμένο url. Ελέγχω αν το status code της αίτησης είναι 200 (επιτυχία). Αν είναι, τότε χρησιμοποιώ το .text για να εξάγω τα περιεχόμενα της απάντησης ως string και στη συνέχεια δημιουργώ έναν reader. Ο reader διαβάζει το αρχείο .csv γραμμή προς γραμμή και το εκτυπώνει (UTF-8). Αν το status code δεν είναι 200, εμφανίζεται κατάλληλο μήνυμα σφάλματος.

```
with requests.Session() as s:
    #download=url
    download = s.get(url)

    """edw diavazw kai typwnw to .csv"""
    if download.status_code == 200:
        data = download.text
        csv_data = csv.reader(data.splitlines())
        for row in csv_data:
            print(row)

    else:
        print("Error: Failed to download CSV data.")
```

Στη συνέχεια χρησιμοποιώ την “utf-8” για να αποκωδικοποιήσω το αρχείο από byte σε string και αποθηκεύω στο decoded\_content.

Δημιουργώ έναν reader που χρησιμοποιεί το decoded\_content ως είσοδο και καθορίζω τον delimiter “,” ώστε να καθορίσω πως το αρχείο είναι .csv χωρισμένο με κόμματα. Έπειτα, με την next(data) παραλείπω την πρώτη γραμμή του .csv, διότι περιλαμβάνει τα headers των στηλών. Μετατρέπω το αρχείο σε μια λίστα data\_list, ώστε να μπορώ να έχω πρόσβαση και χειρισμό στα δεδομένα του.

```
"""anoigma tou csv gia epeksergasia"""
decoded_content = download.content.decode('utf-8')

data = csv.reader(decoded_content.splitlines(), delimiter=',')

#pernaw tin prwti grammi pou exei ta onomata stilwn
next(data)

#vazw ta data se ena list wste na mporw na ta xrisimopoihsa ka
data_list = list(data)
value=float(row[8])
date=row[2]
day = date.split("/")[0]
```

## Ζητούμενο 1:

Χρησιμοποιώ ένα for για να κάνω loop στις γραμμές του data\_list. Για κάθε σειρά εξάγω τον μήνα από την ημερομηνία στην 3η στήλη του .csv χρησιμοποιώντας τη μέθοδο split, ώστε να παίρνω μόνο τον μήνα (DD/MM/YYYY). Στη συνέχεια ορίζω ένα dictionary “indexValue\_per\_month”. Ελέγχω αν η τιμή “month” είναι ήδη κλειδί στο dictionary. Αν είναι, η τιμή του index αυτού του μήνα αυξάνεται κατά value (στήλη 9 του .csv). Αν δεν είναι, δημιουργείται ένα νέο ζεύγος κλειδιού-τιμής στο dictionary, με κλειδί τον μήνα και τιμή το value (στήλη 9 του .csv).

```
indexValue_per_month = {}
```

```

"""Kwdikas gia tziro/month"""
#pernaw kathis grammi tou .csv kai upologizw to value
for row in data_list:
    date = row[2] #date in the third collumn
    month = date.split("/")[1] #pernw ton mina apo tis diafwnes

    if month in indexValue_per_month:
        indexValue_per_month[month] += value
    else:
        indexValue_per_month[month] = value

```

Τώρα που το dictionary μου περιέχει όλα τα ζεύγη κλειδιού-τιμής για τους μήνες, δημιουργώ μια συνάρτηση graph\_value\_per\_month. Όταν αυτή καλείται, εμφανίζεται η bars - παρουσίαση του τζίρου/μήνα. Αρχικά, βρίσκω τη μέγιστη τιμή του dictionary χρησιμοποιώντας τη συνάρτηση max και την αποθηκεύω στη μεταβλητή max\_value. Στη συνέχεια δημιουργώ μια λίστα με τους μήνες που έχουν τη μέγιστη τιμή (max\_value). Αυτό το κάνω, ώστε αν παραπάνω από 2 μήνες έχουν τιμή max\_value, να χρωματιστούν με το ίδιο χρώμα στο γράφημα. Δημιουργώ ένα νέο σχήμα με άξονες χρησιμοποιώντας την plt.subplots και ορίζω το size (10,6). Χρησιμοποιώντας τη μέθοδο ax.bar() με τα κλειδιά του dictionary ως τιμές x και τις τιμές του dictionary ως τιμές y. Στη συνέχεια χρωματίζω τις μπάρες που έχουν τη μέγιστη τιμή μωβ, κι τις υπόλοιπες μπλε (default). Τέλος, δίνω στο γράφημα τίτλο, xlabel, ylabel, υπόμνημα (legend) και το εμφανίζω.

```

"""graph gia tziro/month"""
def graph_value_per_month():

    messagebox.showinfo("Message", "You will see the value/month graph.")
    #vriskw ti megisti timi
    max_value = max(indexValue_per_month.values())

    #vriskw ta transport pou exoun idio max value
    #kanei iterate sta key-value pairs tou indexValuePerMonth kai elegxei an to value einai o max
    #telika krataei megalitero
    max_months = [k for k, v in indexValue_per_month.items() if v == max_value]

    #plots
    fig, ax = plt.subplots(figsize=(10,6))
    bars = ax.bar(indexValue_per_month.keys(), indexValue_per_month.values())

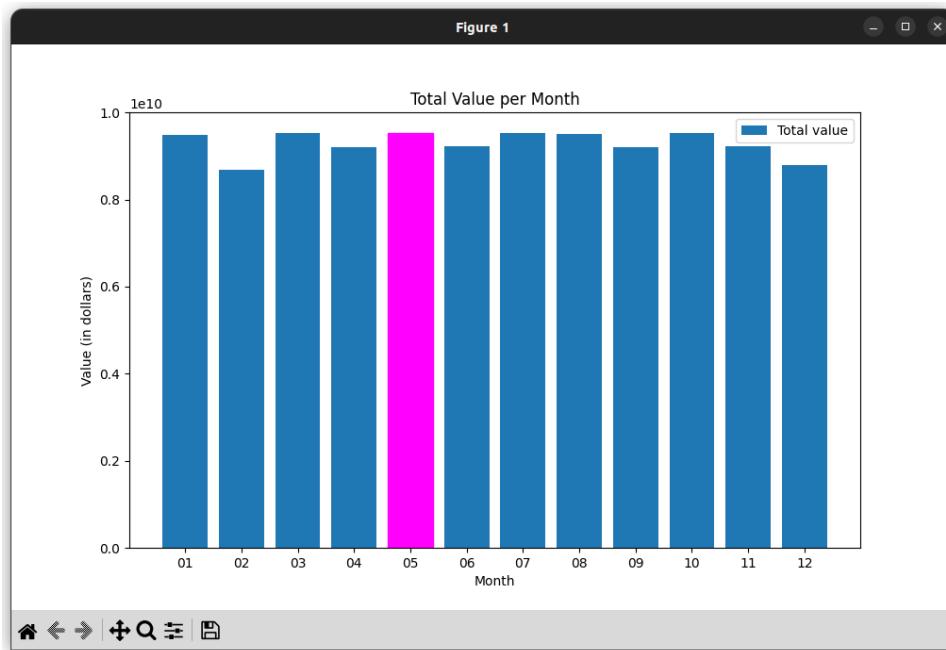
    #idio xrwma sta max values
    for month in max_months:
        bars[list(indexValue_per_month.keys()).index(month)].set_color('magenta')

    #legend
    blue_patch = plt.Rectangle((0, 0), 1, 1, color='blue')
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='magenta')
    plt.legend([blue_patch, magenta_patch], ['Value', 'Max Value'], loc='upper right')

    #first graph: tziros/month
    plt.title("Total Value per Month")
    plt.xlabel("Month")
    plt.ylabel("Value (in dollars)")
    plt.show()

```

Το γράφημα που προκύπτει είναι το εξής:



Παρόμοια μεθοδολογία ακολούθησα και για τα υπόλοιπα ζητούμενα.

## Ζητούμενο 2:

Χρησιμοποιώ ένα for για να κάνω loop στις γραμμές του data\_list. Για κάθε σειρά εξάγω τη χώρα από την 5η στήλη του .csv. Στη συνέχεια ορίζω ένα dictionary “indexValue\_per\_country”. Ελέγχω αν η τιμή “country” είναι ήδη κλειδί στο dictionary. Αν είναι, η τιμή του index αυτής της χώρας αυξάνεται κατά value (στήλη 9 του .csv). Αν δεν είναι, δημιουργείται ένα νέο ζεύγος κλειδιού-τιμής στο dictionary, με κλειδί τη χώρα και τιμή το value (στήλη 9 του .csv).

indexValue\_per\_country={}

```
"""kwdkas gia tziro/country"""
for row in data_list:
    country = row[4] #country in the fifth column

    #xwris to "All" kai to Total
    if country!="All" and country!="Total (excluding China)":
        if country in indexValue_per_country:
            indexValue_per_country[country] += value
        else:
            indexValue_per_country[country] = value
```

Τώρα που το dictionary μου περιέχει όλα τα ζεύγη κλειδιού-τιμής για τις χώρες, δημιουργώ μια συνάρτηση graph\_value\_per\_country. Όταν αυτή καλείται, εμφανίζεται η bars - παρουσίαση του τζίρου/χώρα. Αρχικά, βρίσκω τη μέγιστη τιμή του dictionary χρησιμοποιώντας τη συνάρτηση max και την αποθηκεύω στη μεταβλητή max\_value. Στη συνέχεια δημιουργώ μια λίστα με τις χώρες που έχουν τη μέγιστη τιμή (max\_value). Αυτό το κάνω, ώστε αν παραπάνω από 2 χώρες έχουν τιμή max\_value, να χρωματιστούν με το ίδιο χρώμα στο γράφημα. Δημιουργώ ένα νέο σχήμα με άξονες χρησιμοποιώντας την plt.subplots και ορίζω το size (18,6). Χρησιμοποιώντας τη μέθοδο ax.bar() με τα κλειδιά του dictionary ως τιμές χ και τις τιμές του dictionary ως τιμές γ. Στη συνέχεια χρωματίζω τις μπάρες που έχουν τη μέγιστη τιμή μωβ, κι τις υπόλοιπες μπλε (default). Τέλος, δίνω στο γράφημα τίτλο, xlabel, ylabel, υπόμνημα και το εμφανίζω.

```

"""graph gia tziro/country"""
def graph_value_per_country():

    messagebox.showinfo("Message", "You will see the value/country graph.")
    #vriskw ti megisti timi
    max_value = max(indexValue_per_country.values())

    #vriskw ta transport pou exoun idio max value
    max_countries = [k for k, v in indexValue_per_country.items() if v == max_value]

    #plots
    fig, ax = plt.subplots(figsize=(18,6))
    bars = ax.bar(indexValue_per_country.keys(), indexValue_per_country.values())

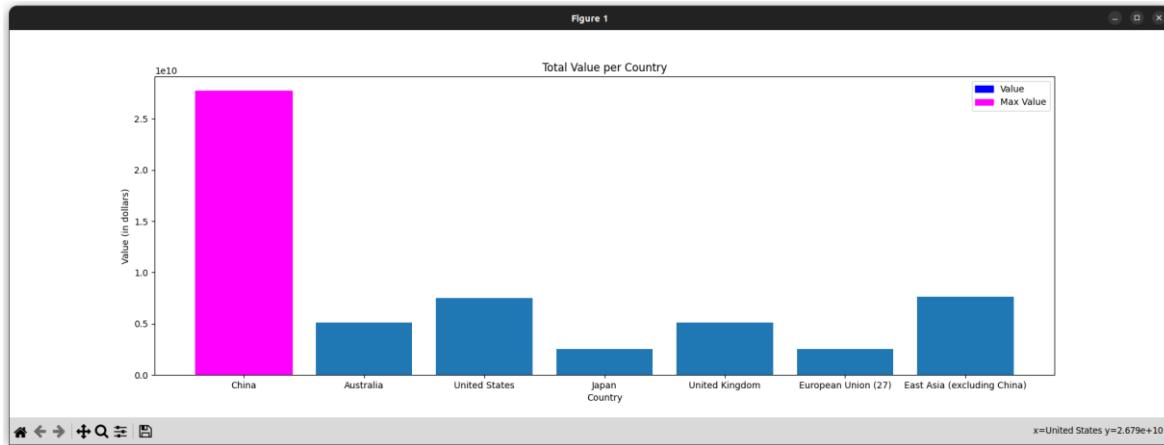
    #idio xrwma st (variable) indexValue_per_country: dict
    for country in max_countries:
        bars[list(indexValue_per_country.keys()).index(country)].set_color('magenta')

    #legend
    blue_patch = plt.Rectangle((0, 0), 1, 1, color='blue')
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='magenta')
    plt.legend([blue_patch, magenta_patch], ['Value', 'Max Value'], loc='upper right')

    #second graph: tziros/country
    plt.title("Total Value per Country")
    plt.xlabel("Country")
    plt.ylabel("Value (in dollars)")
    plt.show()

```

Το γράφημα που προκύπτει είναι το εξής:



### Ζητούμενο 3:

Χρησιμοποιώ ένα for για να κάνω loop στις γραμμές του data\_list. Για κάθε σειρά εξάγω το μέσο μεταφοράς από την 7η στήλη του .csv. Στη συνέχεια ορίζω ένα dictionary “indexValue\_per\_transport”. Ελέγχω αν η τιμή “transport\_mode” είναι ήδη κλειδί στο dictionary. Αν είναι, η τιμή του index αυτού του μέσου μεταφοράς αυξάνεται κατά value (στήλη 9 του .csv). Αν δεν είναι, δημιουργείται ένα νέο ζεύγος

κλειδιού-τιμής στο dictionary, με κλειδί το μέσο μεταφοράς και τιμή το value (στήλη 9 του .csv).

```
indexValue_per_transport={} 
```

```
"""kwdkas gia tziro/transport"""
for row in data_list:
    transport_mode=row[6]

    #xwris to "All"
    if transport_mode!="All":
        if transport_mode in indexValue_per_transport:
            indexValue_per_transport[transport_mode]+=value
        else:
            indexValue_per_transport[transport_mode]=value 
```

Τώρα που το dictionary μου περιέχει όλα τα ζεύγη κλειδιού-τιμής για τα μεταφορικά μέσα, δημιουργώ μια συνάρτηση graph\_value\_per\_transport. Όταν αυτή καλείται, εμφανίζεται η bars - παρουσίαση του τζίρου/μεταφορικό μέσο. Αρχικά, βρίσκω τη μέγιστη τιμή του dictionary χρησιμοποιώντας τη συνάρτηση max και την αποθηκεύω στη μεταβλητή max\_value. Στη συνέχεια δημιουργώ μια λίστα με τα μεταφορικά μέσα που έχουν τη μέγιστη τιμή (max\_value). Αυτό το κάνω, ώστε αν παραπάνω από 2 μεταφορικά μέσα έχουν τιμή max\_value, να χρωματιστούν με το ίδιο χρώμα στο γράφημα. Δημιουργώ ένα νέο σχήμα με άξονες χρησιμοποιώντας την plt.subplots και ορίζω το size (18,6). Χρησιμοποιώντας τη μέθοδο ax.bar() με τα κλειδιά του dictionary ως τιμές x και τις τιμές του dictionary ως τιμές y. Στη συνέχεια χρωματίζω τις μπάρες που έχουν τη μέγιστη τιμή μωβ, κι τις υπόλοιπες μπλε (default). Τέλος, δίνω στο γράφημα τίτλο, xlabel, ylabel, υπόμνημα και το εμφανίζω.

```

"""graph gia tziro/transport"""
def graph_value_per_transport():

    messagebox.showinfo("Message", "You will see the value/transport_mode graph.")
    #vriskw ti megisti timi
    max_value = max(indexValue_per_transport.values())

    #vriskw ta transport pou exoun idio max value
    max_transports = [k for k, v in indexValue_per_transport.items() if v == max_value]

    #plots
    fig, ax = plt.subplots(figsize=(10,6))
    bars = ax.bar(indexValue_per_transport.keys(), indexValue_per_transport.values())

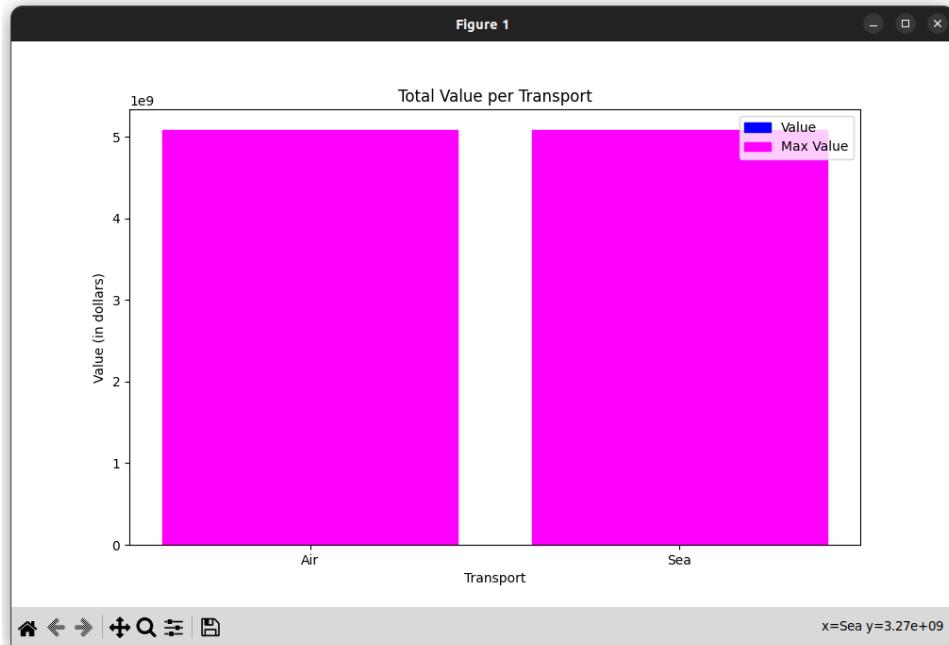
    #idio xrwma sta max values
    for transport in max_transports:
        bars[list(indexValue_per_transport.keys()).index(transport)].set_color('magenta')

    #legend
    blue_patch = plt.Rectangle((0, 0), 1, 1, color='blue')
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='magenta')
    plt.legend([blue_patch, magenta_patch], ['Value', 'Max Value'], loc='upper right')

    #third graph: tziros/transport
    plt.title("Total Value per Transport")
    plt.xlabel("Transport")
    plt.ylabel("Value (in dollars)")
    plt.show()

```

Το γράφημα που προκύπτει είναι το εξής:



#### Ζητούμενο 4:

Χρησιμοποιώ ένα for για να κάνω loop στις γραμμές του data\_list. Για κάθε σειρά εξάγω την ημέρα από την ημερομηνία στην 3η στήλη του .csv χρησιμοποιώντας τη μέθοδο split, ώστε να παίρνω μόνο τη μέρα

(DD/MM/YYYY). Στη συνέχεια ορίζω ένα dictionary “indexValue\_per\_day”. Ελέγχω αν η τιμή “day” είναι ήδη κλειδί στο dictionary. Αν είναι, η τιμή του index αυτής της μέρας αυξάνεται κατά value (στήλη 9 του .csv). Αν δεν είναι, δημιουργείται ένα νέο ζεύγος κλειδιού-τιμής στο dictionary, με κλειδί τη μέρα και τιμή το value (στήλη 9 του .csv).

```
indexValue_per_day={} 
```

```
"""kwdikas gia tziro/day"""
#pernw kai upologisi
for row in data_list:
    date = row[2] #date in the third column
    day = date.split("/")[0] #pernw ti mera

    if day in indexValue_per_day:
        indexValue_per_day[day] += value
    else:
        indexValue_per_day[day] = value 
```

Τώρα που το dictionary μου περιέχει όλα τα ζεύγη κλειδιού-τιμής για τις μέρες, δημιουργώ μια συνάρτηση graph\_value\_per\_day. Όταν αυτή καλείται, εμφανίζεται η bars - παρουσίαση του τζίρου/μέρα. Αρχικά, βρίσκω τη μέγιστη τιμή του dictionary χρησιμοποιώντας τη συνάρτηση max και την αποθηκεύω στη μεταβλητή max\_value. Στη συνέχεια δημιουργώ μια λίστα με τις μέρες που έχουν τη μέγιστη τιμή (max\_value). Αυτό το κάνω, ώστε αν παραπάνω από 2 μέρες έχουν τιμή max\_value, να χρωματιστούν με το ίδιο χρώμα στο γράφημα. Δημιουργώ ένα νέο σχήμα με άξονες χρησιμοποιώντας την plt.subplots και ορίζω το size (12,6). Χρησιμοποιώντας τη μέθοδο ax.bar() με τα κλειδιά του dictionary ως τιμές x και τις τιμές του dictionary ως τιμές y. Στη συνέχεια χρωματίζω τις μπάρες που έχουν τη μέγιστη τιμή μωβ, κι τις υπόλοιπες μπλε (default). Τέλος, δίνω στο γράφημα τίτλο, xlabel, ylabel, υπόμνημα και το εμφανίζω.

```

"""graph gia tziro/day"""
def graph_value_per_day():

    messagebox.showinfo("Message", "You will see the value/day graph.")
    #vriskw ti megisti timi
    max_value = max(indexValue_per_day.values())

    #vriskw ta days pou exoun idio max value
    max_days = [k for k, v in indexValue_per_day.items() if v == max_value]

    #plots
    fig, ax = plt.subplots(figsize=(12,6))
    bars = ax.bar(indexValue_per_day.keys(), indexValue_per_day.values())

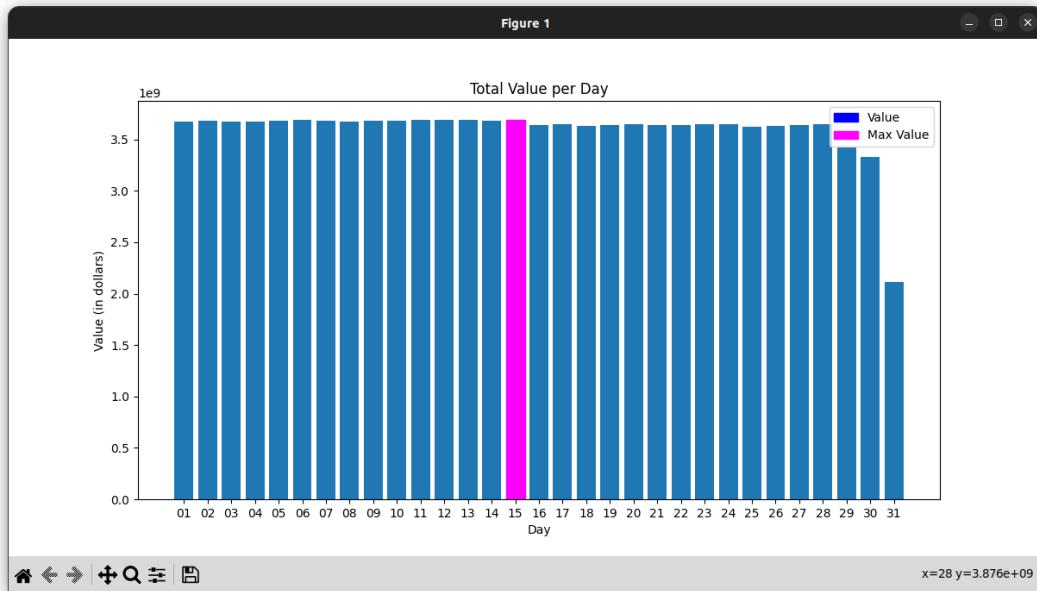
    #idio xrwma sta max values
    for day in max_days:
        bars[list(indexValue_per_day.keys()).index(day)].set_color('magenta')

    #legend
    blue_patch = plt.Rectangle((0, 0), 1, 1, color='blue')
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='magenta')
    plt.legend([blue_patch, magenta_patch], ['Value', 'Max Value'], loc='upper right')

    #forth graph: tziros/transport
    plt.title("Total Value per Day")
    plt.xlabel("Day")
    plt.ylabel("Value (in dollars)")
    plt.show()

```

Το γράφημα που προκύπτει είναι το εξής:



## Ζητούμενο 5:

Χρησιμοποιώ ένα for για να κάνω loop στις γραμμές του data\_list. Για κάθε σειρά εξάγω την κατηγορία εμπορεύματος από την 6η στήλη του .csv. Στη συνέχεια ορίζω ένα dictionary “indexValue\_per\_commodity”. Ελέγχω αν η τιμή “commodity” είναι

ήδη κλειδί στο dictionary. Αν είναι, η τιμή του index αυτής της κατηγορίας εμπορεύματος ανξάνεται κατά value (στήλη 9 του .csv). Αν δεν είναι, δημιουργείται ένα νέο ζεύγος κλειδιού-τιμής στο dictionary, με κλειδί την κατηγορία εμπορεύματος και τιμή το value (στήλη 9 του .csv).

```
indexValue_per_commodity={}
```

```
"""kwikas gia tziro/commodity"""
for row in data_list:
    commodity=row[5]

    #xwris to "All"
    if commodity!="All":
        if commodity in indexValue_per_commodity:
            indexValue_per_commodity[commodity]+=value
        else:
            indexValue_per_commodity[commodity]=value
```

Τώρα που το dictionary μου περιέχει όλα τα ζεύγη κλειδιού-τιμής για τις κατηγορίες εμπορεύματος, δημιουργώ μια συνάρτηση graph\_value\_per\_commodity. Όταν αυτή καλείται, εμφανίζεται η bars - παρουσίαση του τζίρου/κατηγορία εμπορεύματος. Αρχικά, βρίσκω τη μέγιστη τιμή του dictionary χρησιμοποιώντας τη συνάρτηση max και την αποθηκεύω στη μεταβλητή max\_value. Στη συνέχεια δημιουργώ μια λίστα με τις κατηγορίες εμπορεύματος που έχουν τη μέγιστη τιμή (max\_value). Αυτό το κάνω, ώστε αν παραπάνω από 2 κατηγορίες εμπορεύματος έχουν τιμή max\_value, να χρωματιστούν με το ίδιο χρώμα στο γράφημα. Δημιουργώ ένα νέο σχήμα με άξονες χρησιμοποιώντας την plt.subplots και ορίζω το size (18,6). Χρησιμοποιώντας τη μέθοδο ax.bar() με τα κλειδιά του dictionary ως τιμές x και τις τιμές του dictionary ως τιμές y. Στη συνέχεια χρωματίζω τις μπάρες που έχουν τη μέγιστη τιμή μωβ, κι τις υπόλοιπες μπλε (default). Τέλος, δίνω στο γράφημα τίτλο, xlabel, ylabel, υπόμνημα και το εμφανίζω.

```

"""graph gia tziro/commodity"""
def graph_value_per_commodity():

    messagebox.showinfo("Message", "You will see the value/commodity graph.")
    #vriskw ti megisti timi
    max_value = max(indexValue_per_commodity.values())

    #vriskw ta commodities pou exoun idio max value
    max_commodities = [k for k, v in indexValue_per_commodity.items() if v == max_value]

    #plots
    fig, ax = plt.subplots(figsize=(16,10))
    bars = ax.bar(indexValue_per_commodity.keys(), indexValue_per_commodity.values())

    #idio xrwma sta max values
    for commodity in max_commodities:
        bars[list(indexValue_per_commodity.keys()).index(commodity)].set_color('magenta')

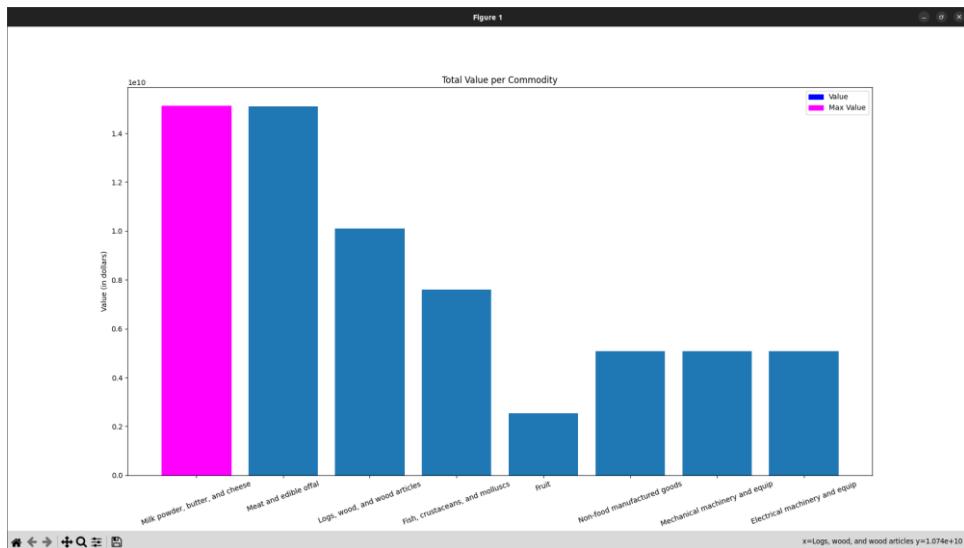
    #vazw klisi sta bar names gia na xwrane
    plt.xticks(rotation=20)

    #legend
    blue_patch = plt.Rectangle((0, 0), 1, 1, color='blue')
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='magenta')
    plt.legend([blue_patch, magenta_patch], ['Value', 'Max Value'], loc='upper right')

    #fifth graph: tziros/transport
    plt.title("Total Value per Commodity")
    plt.xlabel("Commodity")
    plt.ylabel("Value (in dollars)")
    plt.show()

```

Το γράφημα που προκύπτει είναι το εξής:



## Zητούμενο 6:

Για να βρω τους 5 μήνες με τον μεγαλύτερο τζίρο, χρησιμοποιώ το dictionary “indexValue\_per\_month”. Ταξινομώ με φθίνουσα σειρά τα στοιχεία του dictionary με βάση τις τιμές τους και τα αποθηκεύω στη λίστα `sorted_months`. Στη συνέχεια παίρνω τα 5 πρώτα στοιχεία

του sorted\_months, όπου είναι και οι μήνες με το μεγαλύτερο value κι τους αποθηκεύω σε μια λίστα turples top\_5\_months. Δημιουργώ ένα νέο σχήμα με άξονες χρησιμοποιώντας την plt.subplots και ορίζω το size (10,6). Χρησιμοποιώ τη μέθοδο ax.bar, οι τιμές του άξονα χ είναι τα πρώτα στοιχεία (x[0], δηλαδή, οι μήνες) των turples της λίστας top\_5\_months και οι τιμές του άξονα y είναι τα δεύτερα στοιχεία των turples της λίστας top\_5\_months (x[1], δηλαδή, οι συνολικές τιμές). Στη συνέχεια χρωματίζω τις μπάρες που έχουν τη μέγιστη τιμή μωβ, κι τις υπόλοιπες μπλε (default). Τέλος, δίνω στο γράφημα τίτλο, xlabel, ylabel, υπόμνημα και το εμφανίζω.

```
"""graph gia 5 mines me megalitero tziro"""
def graph_5_months():

    messagebox.showinfo("Message", "You will see the 5 most valued months graph.")
    #kanw sort to index gia na parw meta tous megaliterous orous
    #lambda giati apo ta turples pou epistrefontai kratame to deuterio item x[1], dld to \
    sorted_months=sorted(indexValue_per_month.items(), key=lambda x: x[1], reverse=True)

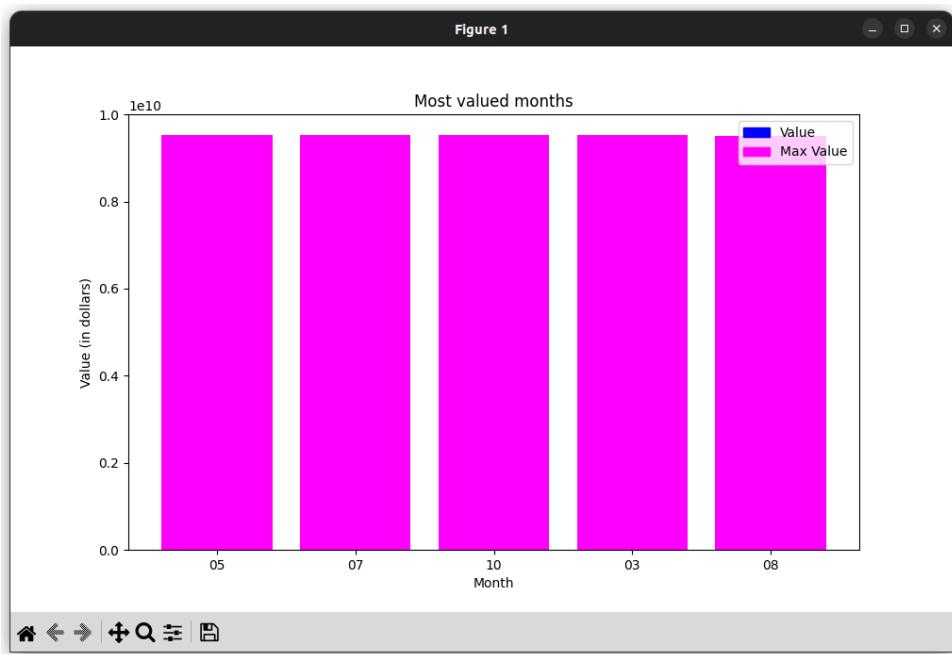
    #pairnw tis 5 times
    top_5_months=sorted_months[:5]

    #plot
    fig, ax = plt.subplots(figsize=(10, 6))
    ax.bar([x[0] for x in top_5_months], [x[1] for x in top_5_months], color='magenta')

    #legend
    blue_patch = plt.Rectangle((0, 0), 1, 1, color='blue')
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='magenta')
    plt.legend([blue_patch, magenta_patch], ['Value', 'Max Value'], loc='upper right')

    #sixth graph
    ax.set_xlabel('Month')
    ax.set_ylabel('Value (in dollars)')
    ax.set_title('Most valued months')
    plt.show()
```

Το γράφημα που προκύπτει είναι το εξής:



### Ζητούμενο 7:

Για την υλοποίηση αυτού του ερωτήματος, δημιουργησα 1 dictionary για κάθε χώρα. Αρχικά, να αναφέρω πως δεν έλαβα υπόψιν μου την περίπτωση “All” του commodity, για αυτό και τα γραφήματα που έφτιαξα αφορούν την Κίνα, τις Ενωμένες Πολιτείες και την Ανατολική Ασία, μιας και αυτές οι περιπτώσεις country είχαν μόνο την περίπτωση “All” για commodity.

```
#dictionaries gia top 5 commodities/country
indexChina={}
indexAustralia={} #exei mono ALL commodity giauto den to vazw
indexUnitedStates={}
indexJapan={} #exei mono ALL commodity giauto den to vazw
indexUnitedKingdom={} #exei mono ALL commodity giauto den to vazw
indexEuropeanUnion={} #exei mono ALL commodity giauto den to vazw
indexEastAsia={} |
```

Χρησιμοποιώ ένα for για να κάνω loop στις γραμμές του data\_list. Για κάθε σειρά εξάγω την κατηγορία εμπορεύματος από την 6η στήλη του .csv και τη χώρα από την 5η σειρά του .csv. Ελέγχω αν η τιμή “commodity” είναι ήδη κλειδί στο dictionary για την εκάστοτε χώρα (παρακάτω θα δείτε κώδικα για όλες τις χώρες). Αν είναι, η τιμή του index αυτής της κατηγορίας εμπορεύματος αυξάνεται κατά value (στήλη 9 του .csv). Αν δεν είναι, δημιουργείται ένα νέο ζεύγος κλειδιού-τιμής στο dictionary, με κλειδί την κατηγορία εμπορεύματος και τιμή το value (στήλη 9 του .csv).

```
"""kwikas for top 5 valued commodities per country CHINA"""
for row in data_list:

    commodity=row[5]
    country=row[4]

    if commodity!="All" and country=="China":
        if commodity in indexChina:
            indexChina[commodity]+=value
        else:
            indexChina[commodity]=value
```

```
"""kwikas for top 5 valued commodities per country UN
for row in data_list:

    commodity=row[5]
    country=row[4]

    if commodity!="All" and country=="United States":
        if commodity in indexUnitedStates:
            indexUnitedStates[commodity]+=value
        else:
            indexUnitedStates[commodity]=value
```

```
"""kwikas for top 5 valued commodities per country ASIA"""
for row in data_list:
    commodity=row[5]
    country=row[4]
    if commodity!="All" and country=="East Asia (excluding China)":
        if commodity in indexEastAsia:
            indexEastAsia[commodity]+=value
        else:
            indexEastAsia[commodity]=value
```

Ταξινομώ με φθίνουσα σειρά τα στοιχεία του εκάστοτε dictionary με βάση τις τιμές τους και τα αποθηκεύω στη λίστα turples sorted\_commodities. Στη συνέχεια παίρνω τα 5 πρώτα στοιχεία του sorted\_commodities, όπου είναι και οι κατηγορίες εμπορεύματος με το μεγαλύτερο value κι τις αποθηκεύω σε μια λίστα turples top\_5\_commodities. Δημιουργώ ένα νέο σχήμα με άξονες χρησιμοποιώντας την plt.subplots και ορίζω το size (18,6) (για την Κίνα). Χρησιμοποιώ τη μέθοδο ax.bar, οι τιμές του άξονα χ είναι τα πρώτα στοιχεία (x[0], δηλαδή, οι κατηγορίες εμπορεύματος) των turples της λίστας top\_5\_commodities και οι τιμές του άξονα y είναι τα δεύτερα στοιχεία των turples της λίστας top\_5\_commodities (x[1], δηλαδή, οι συνολικές τιμές). Στη συνέχεια χρωματίζω τις μπάρες με κόκκινο χρώμα. Τέλος, δίνω στο γράφημα τίτλο, xlabel, ylabel, υπόμνημα και το εμφανίζω.

```
"""graphs via top 5 valued commodities/country"""
def graph_5_commodities_china():

    messagebox.showinfo("Message", "You will see the graph about the 5 most valued commodities")

    sorted_commodities_china=sorted(indexChina.items(), key=lambda x: x[1], reverse=True)
    top_5_commodities=sorted_commodities_china[:5]

    fig, ax = plt.subplots(figsize=(18, 6))
    ax.bar([x[0] for x in top_5_commodities], [x[1] for x in top_5_commodities], color='red')

    #legend
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
    plt.legend([magenta_patch], ['Value'], loc='upper right')

    ax.set_xlabel('Commodities')
    ax.set_ylabel('China')
    ax.set_title('Most valued commodities')
    plt.show()
```

```
def graph_5_commodities_US():

    messagebox.showinfo("Message", "You will see the graph about the 5 most valued commodities for US.")

    sorted_commodities_US=sorted(indexUnitedStates.items(), key=lambda x: x[1], reverse=True)
    top_5_commodities=sorted_commodities_US[:5]

    fig, ax = plt.subplots(figsize=(4, 6))
    ax.bar([x[0] for x in top_5_commodities], [x[1] for x in top_5_commodities], color='red')

    #legend
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
    plt.legend([magenta_patch], ['Value'], loc='upper right')

    ax.set_xlabel('Commodities')
    ax.set_ylabel('United States')
    ax.set_title('Most valued commodities')
    plt.show()
```

```

def graph_5_commodities_EA():
    messagebox.showinfo("Message", "You will see the graph about the 5 most valued commodities for EA.")

    sorted_commodities_ea=sorted(indexEastAsia.items(), key=lambda x: x[1], reverse=True)
    top_5_commodities=sorted_commodities_ea[:5]

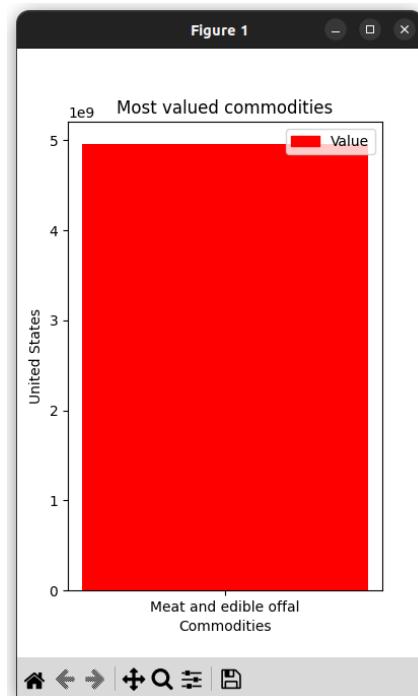
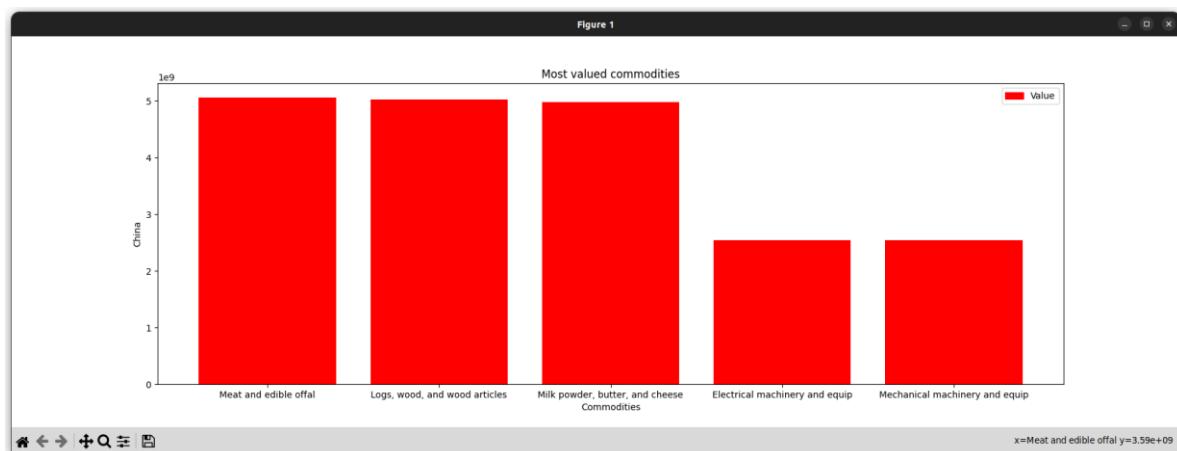
    fig, ax = plt.subplots(figsize=(4, 6))
    ax.bar([x[0] for x in top_5_commodities], [x[1] for x in top_5_commodities], color='red')

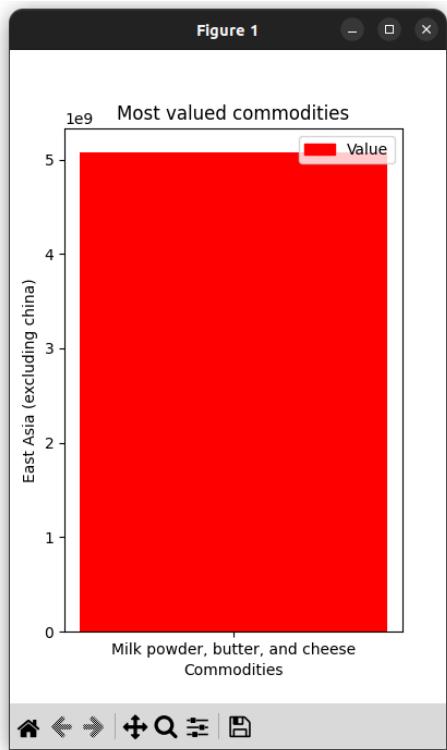
    #legend
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
    plt.legend([magenta_patch], ['Value'], loc='upper right')

    ax.set_xlabel('Commodities')
    ax.set_ylabel('East Asia (excluding china)')
    ax.set_title('Most valued commodities')
    plt.show()

```

Τα γραφήματα που προκύπτουν είναι τα εξής:





## Ζητούμενο 8:

Για την υλοποίηση αυτού του ερωτήματος, δημιουργησα 1 dictionary για κάθε κατηγορία εμπορεύματος, με κλειδιά τις ημέρες της εβδομάδας και αρχικοποίηση τιμών 0. Να αναφέρω, επίσης, πως δεν έλαβα υπόψιν μου την περίπτωση “All” του commodity.

```
#indexes gia top day/commodity
milk = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
meat = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
logs = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
fish = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
fruit = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
non_food = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
mechanical = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
electrical = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
```

Χρησιμοποιώ ένα for για να κάνω loop στις γραμμές του data\_list. Για κάθε σειρά εξάγω την κατηγορία εμπορεύματος από την 6η στήλη του .csv και τη μέρα από την 4η σειρά του .csv. Ελέγχω αν η τιμή “weekday” είναι ήδη κλειδί στο dictionary για την εκάστοτε κατηγορία εμπορεύματος (παρακάτω θα δείτε κώδικα για όλες τις κατηγορίες εμπορεύματος). Αν είναι, η τιμή του index αυτής της μέρας αυξάνεται

κατά value (στήλη 9 του .csv). Αν δεν είναι, η τιμή του ζεύγους κλειδιού-τιμής στο dictionary παραμένει με τιμή 0.

```
"""top days for milk"""
for row in data_list:
    commodity=row[5]
    weekday=row[3]

    if commodity=="Milk powder, butter, and cheese":
        if weekday in milk:
            milk[weekday]+=value
        else:
            milk[weekday]=value
```

```
"""top days for meat"""
for row in data_list:
    commodity=row[5]
    weekday=row[3]

    if commodity=="Meat and edible offal":
        if weekday in meat:
            meat[weekday]+=value
        else:
            meat[weekday]=value
```

```
"""top days for logs"""
for row in data_list:
    commodity=row[5]
    weekday=row[3]

    if commodity=="Logs, wood, and wood articles":
        if weekday in logs:
            logs[weekday]+=value
        else:
            logs[weekday]=value
```

```
"""top days for fish"""
for row in data_list:
    commodity=row[5]
    weekday=row[3]

    if commodity=="Fish, crustaceans, and molluscs":
        if weekday in fish:
            fish[weekday]+=value
        else:
            fish[weekday]=value
```

```

"""top days for fruit"""
for row in data_list:
    commodity=row[5]
    weekday=row[3]

    if commodity=="Fruit":
        if weekday in fruit:
            fruit[weekday]+=value
        else:
            fruit[weekday]=value

```

```

"""top days for non food"""
for row in data_list:
    commodity=row[5]
    weekday=row[3]

    if commodity=="Non-food manufactured goods":
        if weekday in fruit:
            non_food[weekday]+=value
        else:
            non_food[weekday]=value

```

```

"""top days for mechanical"""
for row in data_list:
    commodity=row[5]
    weekday=row[3]

    if commodity=="Mechanical machinery and equip":
        if weekday in fruit:
            mechanical[weekday]+=value
        else:
            mechanical[weekday]=value

```

```

"""top days for electrical"""
for row in data_list:
    commodity=row[5]
    weekday=row[3]

    if commodity=="Electrical machinery and equip":
        if weekday in fruit:
            electrical[weekday]+=value
        else:
            electrical[weekday]=value

```

Ταξινομώ με φθίνουσα σειρά τα στοιχεία του εκάστοτε dictionary με βάση τις τιμές τους και τα αποθηκεύω στη λίστα turples sorted\_days.

Στη συνέχεια παίρνω τα 5 πρώτα στοιχεία του sorted\_days, όπου είναι και οι μέρες με το μεγαλύτερο value κι τις αποθηκεύω σε μια λίστα turples top\_5\_days. Δημιουργώ ένα νέο σχήμα με άξονες χρησιμοποιώντας την plt.subplots και ορίζω το size (5,6) (για την κατηγορία εμπορεύματος γάλα...). Χρησιμοποιώ τη μέθοδο ax.bar, οι τιμές του άξονα χίναι τα πρώτα στοιχεία (x[0], δηλαδή, οι μέρες) των turples της λίστας top\_5\_days και οι τιμές του άξονα y είναι τα δεύτερα στοιχεία των turples της λίστας top\_5\_days (x[1], δηλαδή, οι συνολικές τιμές). Στη συνέχεια χρωματίζω τις μπάρες με κόκκινο χρώμα. Τέλος, δίνω στο γράφημα τίτλο, xlabel, ylabel, υπόμνημα και το εμφανίζω.

```
"""graphs για top day/commodity"""
def top_milk():

    messagebox.showinfo("Message", "You will see the graph about the most valued day for milk powder, butter, and cheese.")

    sorted_days=sorted(milk.items(), key=lambda x: x[1], reverse=True)
    top5=sorted_days[:1]

    fig, ax = plt.subplots(figsize=(5, 6))
    ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')

    #Legend
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
    plt.legend([magenta_patch], ['Value'], loc='upper right')

    ax.set_xlabel('Day')
    ax.set_ylabel('Milk powder, butter, and cheese')
    ax.set_title('Most valued day')
    plt.show()
```

```
def top_meat():

    messagebox.showinfo("Message", "You will see the graph about the most valued day for meat and edible offal")

    sorted_days=sorted(meat.items(), key=lambda x: x[1], reverse=True)
    top5=sorted_days[:1]

    fig, ax = plt.subplots(figsize=(5, 6))
    ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')

    #legend
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
    plt.legend([magenta_patch], ['Value'], loc='upper right')

    ax.set_xlabel('Day')
    ax.set_ylabel('Meat and edible offal')
    ax.set_title('Most valued day')
    plt.show()
```

```

def top_logs():

    messagebox.showinfo("Message", "You will see the graph about the most valued day for logs, wood, and wood articles")

    sorted_days=sorted(logs.items(), key=lambda x: x[1], reverse=True)
    top5=sorted_days[:1]

    fig, ax = plt.subplots(figsize=(5, 6))
    ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')

    #legend
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
    plt.legend([magenta_patch], ['Value'], loc='upper right')

    ax.set_xlabel('Day')
    ax.set_ylabel('Logs, wood, and wood articles')
    ax.set_title('Most valued day')
    plt.show()

```

```

def top_fish():

    messagebox.showinfo("Message", "You will see the graph about the most valued day for fish, crustaceans, and molluscs")

    sorted_days=sorted(fish.items(), key=lambda x: x[1], reverse=True)
    top5=sorted_days[:1]

    fig, ax = plt.subplots(figsize=(5, 6))
    ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')

    #legend
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
    plt.legend([magenta_patch], ['Value'], loc='upper right')

    ax.set_xlabel('Day')
    ax.set_ylabel('Fish, crustaceans, and molluscs')
    ax.set_title('Most valued day')
    plt.show()

```

```

def top_fruit():

    messagebox.showinfo("Message", "You will see the graph about the most valued day for fruit")

    sorted_days=sorted(fruit.items(), key=lambda x: x[1], reverse=True)
    top5=sorted_days[:1]
        (function) figsize: tuple[Literal[5], Literal[6]]
    fig, ax = plt.subplots(figsize=(5, 6))
    ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')

    #legend
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
    plt.legend([magenta_patch], ['Value'], loc='upper right')

    ax.set_xlabel('Day')
    ax.set_ylabel('Fruit')
    ax.set_title('Most valued day')
    plt.show()

```

```

def top_non_food():

    messagebox.showinfo("Message", "You will see the graph about the most valued day for non-food manufactured goods")

    sorted_days=sorted(non_food.items(), key=lambda x: x[1], reverse=True)
    top5=sorted_days[:1]

    fig, ax = plt.subplots(figsize=(5, 6))
    ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')

    #legend
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
    plt.legend([magenta_patch], ['Value'], loc='upper right')

    ax.set_xlabel('Day')
    ax.set_ylabel('Non-food manufactured goods')
    ax.set_title('Most valued day')
    plt.show()

```

```

def top_mechanical():

    messagebox.showinfo("Message", "You will see the graph about the most valued day for mechanical machinery and equip")

    sorted_days=sorted(mechanical.items(), key=lambda x: x[1], reverse=True)
    top5=sorted_days[:1]

    fig, ax = plt.subplots(figsize=(5, 6))
    ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')

    #legend
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
    plt.legend([magenta_patch], ['Value'], loc='upper right')

    ax.set_xlabel('Day')
    ax.set_ylabel('Mechanical machinery and equip')
    ax.set_title('Most valued day')
    plt.show()

```

```

def top_electrical():

    messagebox.showinfo("Message", "You will see the graph about the most valued day for electrical machinery and equip")

    sorted_days=sorted(electrical.items(), key=lambda x: x[1], reverse=True)
    top5=sorted_days[:1]

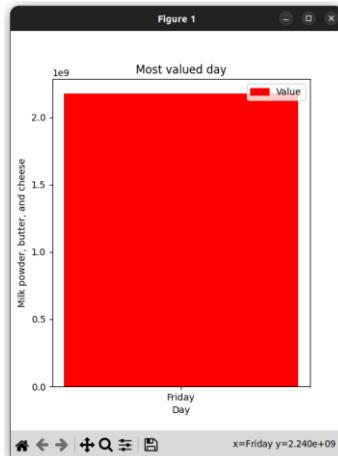
    fig, ax = plt.subplots(figsize=(5, 6))
    ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')

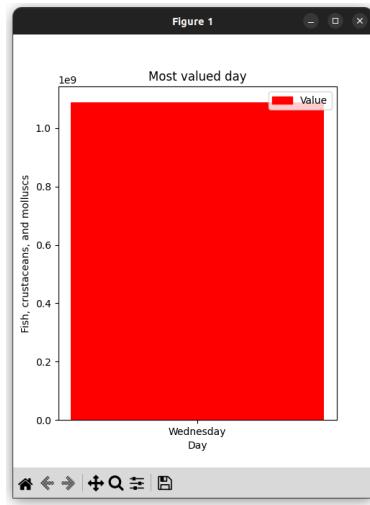
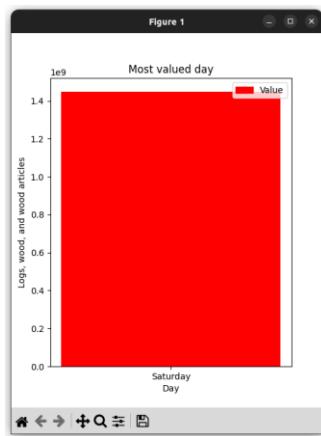
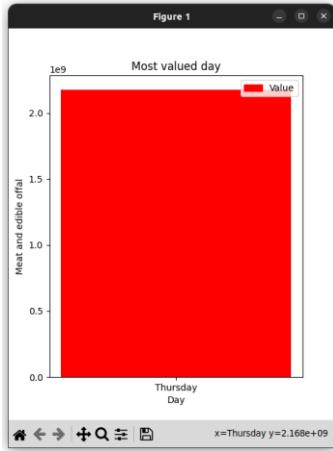
    #legend
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
    plt.legend([magenta_patch], ['Value'], loc='upper right')

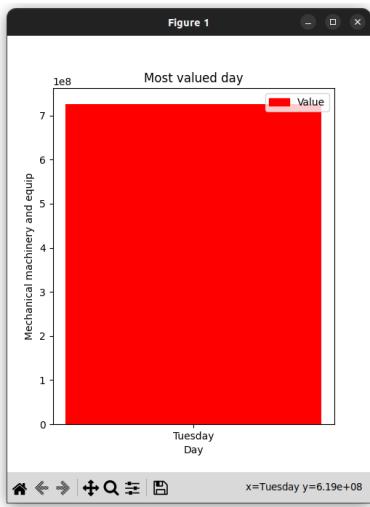
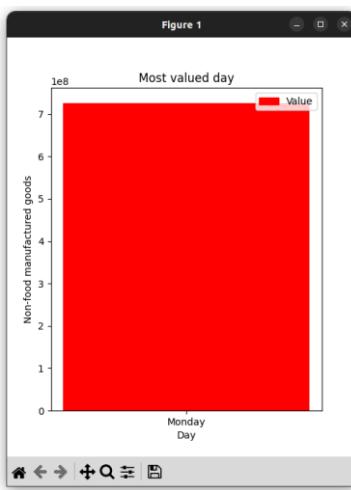
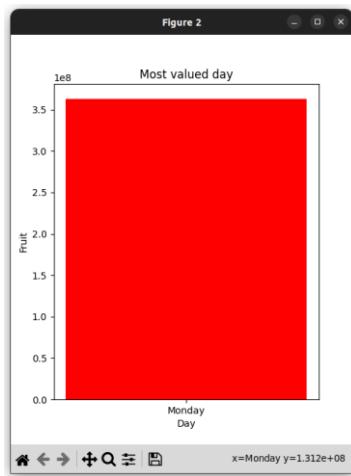
    ax.set_xlabel('Day')
    ax.set_ylabel('Electrical machinery and equip')
    ax.set_title('Most valued day')
    plt.show()

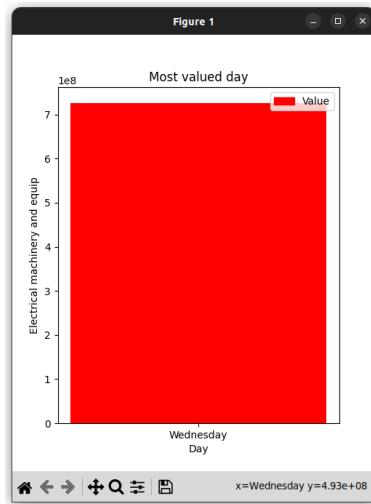
```

Τα γραφήματα που προκύπτουν είναι τα εξής:









### C) SQL, CSV κώδικας:

Για να εξάγω τα δεδομένα που παράγουν οι συναρτήσεις μου σε πίνακες SQL, αρχικά δημιουργώ σύνδεση με τη βάση που έχω δημιουργήσει τοπικά στον υπολογιστή μου με όνομα covid\_effects.

```
#sindei sto database
cnx = mysql.connector.connect(user='root', password='Melenegiorgo2002 ', host='localhost', database='covid_effects')
```

Εντός κάθε συνάρτησης που έχω δημιουργήσει για να εξάγω τα αποτελέσματα των ζητούμενων, δημιουργώ έναν cursor πάνω στη σύνδεση μου με την sql βάση, ώστε να εκτελώ sql εντολές με αυτόν. Στη συνέχεια δημιουργώ ένα table όπου και θα εισάγω τα εκάστοτε δεδομένα (το κάνω και drop αν υπάρχει ήδη ώστε να γίνεται ανανέωση). Στη συνέχεια με ένα for loop παίρνω τα κλειδιά και τις τιμές του εκάστοτε dictionary και τα βάζω στον εκάστοτε πίνακα. Τέλος, κάνω commit τις αλλαγές ώστε να κρατηθούν.

Όσον αφορά την εξαγωγή .csv αρχείων με τα αποτελέσματα, δημιουργώ .csv αρχεία στον φάκελο του project και τα ανοίγω με

δικαίωμα “write”. Στη συνέχεια, φτιάχνω έναν writer, με τον οποίο θα γράφω δεδομένα στο εκάστοτε .csv αρχείο και εκτελώ μια εντολή sql “select \* from table \_name”, ώστε να πάρω τα δεδομένα από τον πίνακα και να τα αποθηκεύσω. Κρατώ την γραμμή i[0] για τα ονόματα των στηλών του .csv και στη συνέχεια γράφω τις υπόλοιπες γραμμές που φέρνει ο cursor στο .csv αρχείο.

Τέλος, κλείνω τον cursor.

Ιδια λογική ακολουθώ εντός όλων των συναρτήσεων που θέλω να κρατήσω τα αποτελέσματα τους σε sql πίνακες και .csv αρχεία.

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS ValuePerMonth")
cursor.execute("CREATE TABLE ValuePerMonth (month INT, indexValue BIGINT)")

for month, indexValue in indexValue_per_month.items():
    cursor.execute("INSERT INTO ValuePerMonth VALUES (%s, %s)", (month, indexValue))

cnx.commit()

#gia .csv file
with open('ValuePerMonth.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM ValuePerMonth")
    #gia ta onoma twn stilwn
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()
```

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS ValuePerCountry")
cursor.execute("CREATE TABLE ValuePerCountry (country VARCHAR(255), indexValue BIGINT)")

for country, indexValue in indexValue_per_country.items():
    cursor.execute("INSERT INTO ValuePerCountry VALUES (%s, %s)", (country, indexValue))

cnx.commit()
```

```
#gia .csv file
with open('ValuePerCountry.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM ValuePerCountry")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()
```

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS ValuePerTransport")
cursor.execute("CREATE TABLE ValuePerTransport (transport VARCHAR(255), indexValue BIGINT)")

for transport, indexValue in indexValue_per_transport.items():
    cursor.execute("INSERT INTO ValuePerTransport VALUES (%s, %s)", (transport, indexValue))

cnx.commit()

#gia .csv file
with open('ValuePerTransport.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM ValuePerTransport")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()
```

```

#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS ValuePerDay")
cursor.execute("CREATE TABLE ValuePerDay (day INT, indexValue BIGINT)")

for day, indexValue in indexValue_per_day.items():
    cursor.execute("INSERT INTO ValuePerDay VALUES (%s, %s)", (day, indexValue))

cnx.commit()

#gia .csv file
with open('ValuePerDay.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM ValuePerDay")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()

```

```

#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS ValuePerCommodity")
cursor.execute("CREATE TABLE ValuePerCommodity (commodity VARCHAR(255), indexValue BIGINT)")

for commodity, indexValue in indexValue_per_transport.items():
    cursor.execute("INSERT INTO ValuePerCommodity VALUES (%s, %s)", (commodity, indexValue))

cnx.commit()

#gia .csv file
with open('ValuePerCommodity.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM ValuePerCommodity")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()

```

```

#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopFiveMonths")
cursor.execute("CREATE TABLE TopFiveMonths (month INT, indexValue BIGINT)")

for month, indexValue in top_5_months:
    cursor.execute("INSERT INTO TopFiveMonths VALUES (%s, %s)", (month, indexValue))

cnx.commit()

#gia .csv file
with open('TopFiveMonths.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopFiveMonths")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()

```

```

#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopCommoditiesChina")
cursor.execute("CREATE TABLE TopCommoditiesChina (commodity VARCHAR(255), indexValue DOUBLE)")

for commodity, indexValue in top_5_commodities:
    cursor.execute("INSERT INTO TopCommoditiesChina VALUES (%s, %s)", (commodity, indexValue))

cnx.commit()

#gia .csv file
with open('TopCommoditiesChina.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopCommoditiesChina")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()

```

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopCommoditiesUS")
cursor.execute("CREATE TABLE TopCommoditiesUS (commodity VARCHAR(255), indexValue DOUBLE)")

for commodity, indexValue in top_5_commodities:
    cursor.execute("INSERT INTO TopCommoditiesUS VALUES (%s, %s)", (commodity, indexValue))

cnx.commit()

#gia .csv file
with open('TopCommoditiesUS.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopCommoditiesUS")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()
```

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopCommoditiesEA")
cursor.execute("CREATE TABLE TopCommoditiesEA (commodity VARCHAR(255), indexValue DOUBLE)")

for commodity, indexValue in top_5_commodities:
    cursor.execute("INSERT INTO TopCommoditiesEA VALUES (%s, %s)", (commodity, indexValue))

cnx.commit()

#gia .csv file
with open('TopCommoditiesEA.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopCommoditiesEA")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()
```

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopDayForMilk")
cursor.execute("CREATE TABLE TopDayForMilk (commodity VARCHAR(255), indexValue DOUBLE)")

for commodity, indexValue in top5:
    cursor.execute("INSERT INTO TopDayForMilk VALUES (%s, %s)", (commodity, indexValue))

cnx.commit()

#gia .csv file
with open('TopDayForMilk.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopDayForMilk")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()
```

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopDayForMeat")
cursor.execute("CREATE TABLE TopDayForMeat (commodity VARCHAR(255), indexValue DOUBLE)")

for commodity, indexValue in top5:
    cursor.execute("INSERT INTO TopDayForMeat VALUES (%s, %s)", (commodity, indexValue))

cnx.commit()

#gia .csv file
with open('TopDayForMeat.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopDayForMeat")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()
```

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopDayForWood")
cursor.execute("CREATE TABLE TopDayForWood (commodity VARCHAR(255), indexValue DOUBLE)")

for commodity, indexValue in top5:
    cursor.execute("INSERT INTO TopDayForWood VALUES (%s, %s)", (commodity, indexValue))

cnx.commit()

#gia .csv file
with open('TopDayForWood.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopDayForWood")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()
```

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopDayForFish")
cursor.execute("CREATE TABLE TopDayForFish (commodity VARCHAR(255), indexValue DOUBLE)")

for commodity, indexValue in top5:
    cursor.execute("INSERT INTO TopDayForFish VALUES (%s, %s)", (commodity, indexValue))

cnx.commit()

#gia .csv file
with open('TopDayForFish.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopDayForFish")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()
```

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopDayForFruit")
cursor.execute("CREATE TABLE TopDayForFruit (commodity VARCHAR(255), indexValue DOUBLE)")

for commodity, indexValue in top5:
    cursor.execute("INSERT INTO TopDayForFruit VALUES (%s, %s)", (commodity, indexValue))

cnx.commit()

#gia .csv file
with open('TopDayForFruit.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopDayForFruit")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()
```

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopDayForNonFood")
cursor.execute("CREATE TABLE TopDayForNonFood (commodity VARCHAR(255), indexValue DOUBLE)")

for commodity, indexValue in top5:
    cursor.execute("INSERT INTO TopDayForNonFood VALUES (%s, %s)", (commodity, indexValue))

cnx.commit()

#gia .csv file
with open('TopDayForNonFood.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopDayForNonFood")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()
```

```

#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopDayMechanical")
cursor.execute("CREATE TABLE TopDayMechanical (commodity VARCHAR(255), indexValue DOUBLE)")

for commodity, indexValue in top5:
    cursor.execute("INSERT INTO TopDayMechanical VALUES (%s, %s)", (commodity, indexValue))

cnx.commit()

#gia .csv file
with open('TopDayMechanical.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopDayMechanical")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()

#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopDayElectrical")
cursor.execute("CREATE TABLE TopDayElectrical (commodity VARCHAR(255), indexValue DOUBLE)")

for commodity, indexValue in top5:
    cursor.execute("INSERT INTO TopDayElectrical VALUES (%s, %s)", (commodity, indexValue))

cnx.commit()

#gia .csv file
with open('TopDayElectrical.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopDayElectrical")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()

```

Στη συνέχεια αν εκτελέσω “show tables;” στη βάση μου:

```

mysql> show tables;
+-----+
| Tables_in_covid_effects |
+-----+
| TopCommoditiesChina
| TopCommoditiesEA
| TopCommoditiesUS
| TopDayElectrical
| TopDayForFish
| TopDayForFruit
| TopDayForMeat
| TopDayForMilk
| TopDayForNonFood
| TopDayForWood
| TopDayMechanical
| TopFiveMonths
| ValuePerCommodity
| ValuePerCountry
| ValuePerDay
| ValuePerMonth
| ValuePerTransport
+-----+
17 rows in set (0,00 sec)

mysql>

```

Και κάθε πίνακας περιέχει τα αποτελέσματα του εκάστοτε ζητούμενου.  
Για παράδειγμα, ο πίνακας ValuePerMonth:

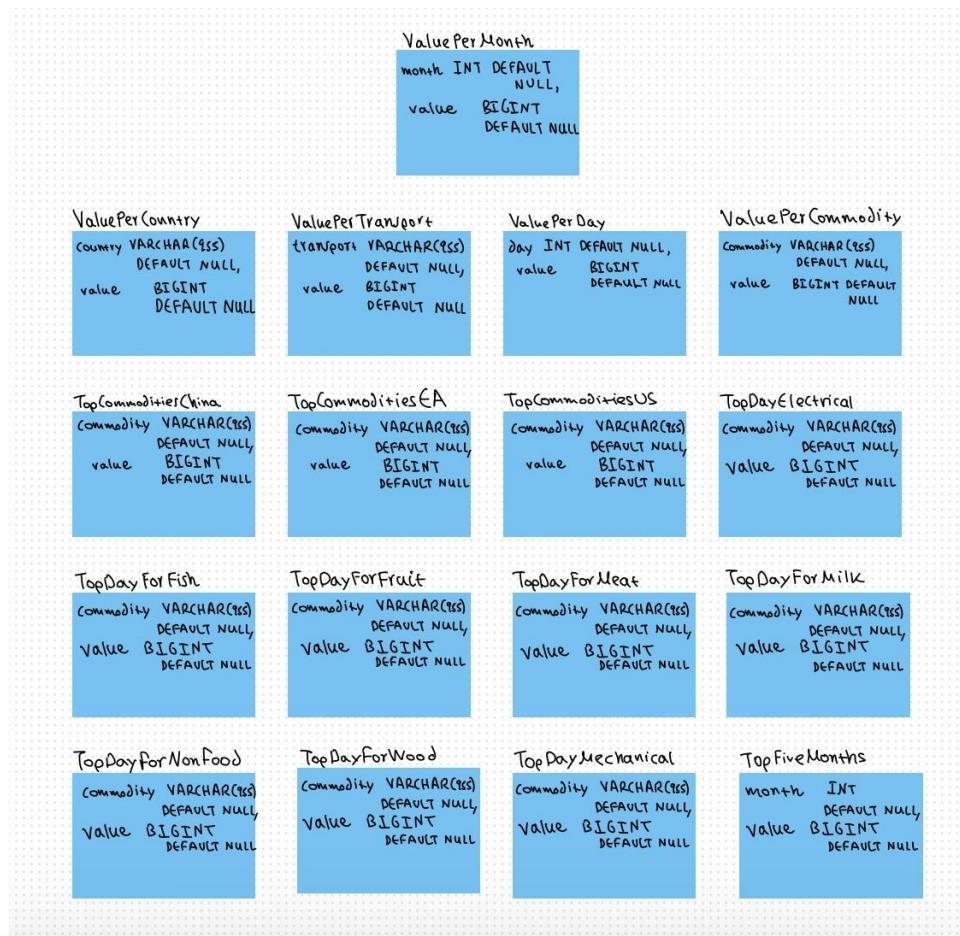
```

mysql> select * from ValuePerMonth;
+-----+-----+
| month | indexValue |
+-----+-----+
| 1     | 9492000000 |
| 2     | 8681000000 |
| 3     | 9521000000 |
| 4     | 9205000000 |
| 5     | 9530000000 |
| 6     | 9217000000 |
| 7     | 9528000000 |
| 8     | 9513000000 |
| 9     | 9203000000 |
| 10    | 9524000000 |
| 11    | 9228000000 |
| 12    | 8796000000 |
+-----+-----+
12 rows in set (0,00 sec)

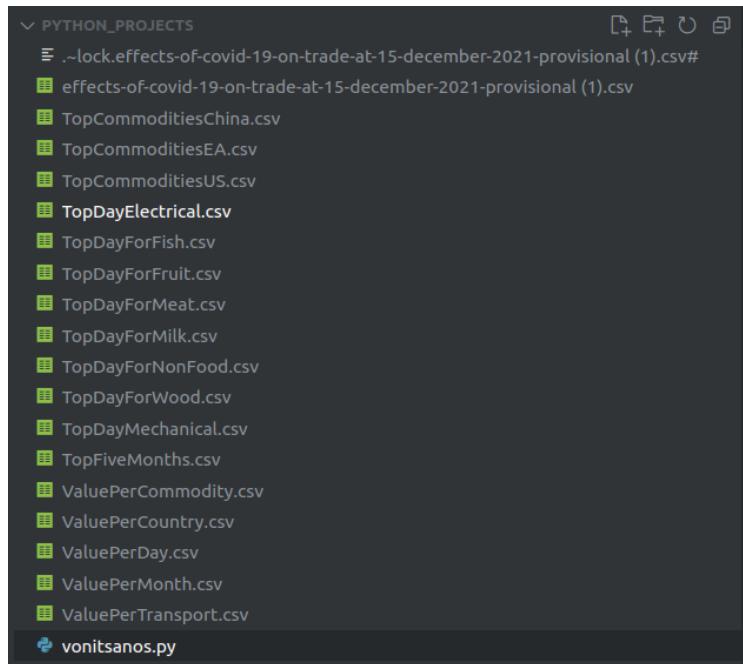
mysql> 

```

Το τελικό ER Model της βάσης μου είναι το εξής:



Όσον αφορά τα .csv αρχεία που εξάγονται:



Κάθε ένα περιέχει τα αποτελέσματα του εκάστοτε ζητούμενο. Για παράδειγμα το ValuePerMonth.csv:

```
vonitsanos.py ValuePerMonth.csv
ValuePerMonth.csv
1 month,indexValue
2 1,9492000000
3 2,8681000000
4 3,9521000000
5 4,9205000000
6 5,9530000000
7 6,9217000000
8 7,9528000000
9 8,9513000000
10 9,9203000000
11 10,9524000000
12 11,9228000000
13 12,8796000000
14
```

The screenshot shows a code editor with two tabs: 'vonitsanos.py' and 'ValuePerMonth.csv'. The 'ValuePerMonth.csv' tab is active and displays the following data:

month	indexValue
1	1,9492000000
2	2,8681000000
3	3,9521000000
4	4,9205000000
5	5,9530000000
6	6,9217000000
7	7,9528000000
8	8,9513000000
9	9,9203000000
10	10,9524000000
11	11,9228000000
12	12,8796000000
13	
14	

## D) Graphical User Interface (GUI):

Αποφάσισα στο project μου να δημιουργήσω ένα μενού με κουμπιά, όπου κάθε κουμπί καλεί μια συνάρτηση που εμφανίζει κάποιο γράφημα. Χρησιμοποίησα τη βιβλιοθήκη tkinter για τη δημιουργία των frames και των buttons.

Δημιουργώ ένα root frame με τίτλο “Covid Effects on Trades”. Στη συνέχεια δημιουργώ ένα main frame εντός του root. Εντός του main frame δημιουργώ 3 subframes, ώστε να χωρέσω όμορφα τα buttons που θα χρησιμοποιήσω.

Στη συνέχεια φτιάχνω τα κουμπιά, όπου το καθένα καλεί μια συνάρτηση (που μας παρουσιάζει το εκάστοτε γράφημα).

Επίσης, όσον αφορά το ζητούμενο 7 και 8, έχω δημιουργήσει button που μας πετάει σε νέο frame, όπου εμφανίζονται νέα buttons. Αυτό το πέτυχα με 2 συναρτήσεις: new\_window\_for\_commodities, new\_window\_for\_days, οι οποίες όταν καλούνται ανοίγουν καινούριο frame με buttons. Επέλεξα αυτή την υλοποίηση, διότι έχω χωρίσει τα γραφήματα στο ζητούμενο 7 ανά κατηγορία εμπορεύματος και στο ζητούμενο 8 ανά μέρα.

```
"""gia na anoigw new window gia epilogi country pou psaxnw gia top 5 valued commodities"""
def new_window_for_commodities():
    new_window = tk.Toplevel(root)
    new_window.title("Top Commodities/Country")
    new_window.configure(background='blue')
    new_window.geometry("740x200")

    button_china= tk.Button(new_window, text="China", command=graph_5_commodities_china, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_china.pack(side=tk.LEFT, padx=10)

    button_US= tk.Button(new_window, text="United States", command=graph_5_commodities_US, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_US.pack(side=tk.LEFT, padx=10)

    button_EA= tk.Button(new_window, text="East Asia", command=graph_5_commodities_EA, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_EA.pack(side=tk.LEFT, padx=10)
```

```

"""dia na anoiwg new window gia epilogi commodity pay psaxnw gia top valued day"""
def new_window_for_days():
    new_window = tk.Toplevel(root)
    new_window.title("Top Commodities/Country")
    new_window.configure(background='blue')
    new_window.geometry('1000x150')

    #panel 1-button frame 1 dia ta panw 4 koumpia
    button_frame1 = tk.Frame(new_window, bg='blue')
    button_frame1.pack(side='top', fill='both', expand=True, padx=10, pady=10)

    button_milk= tk.Button(button_frame1, text="Milk", command=top_milk, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_milk.pack(side=tk.LEFT, padx=10)

    button_meat= tk.Button(button_frame1, text="Meat", command=top_meat, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_meat.pack(side=tk.LEFT, padx=10)

    button_logs= tk.Button(button_frame1, text="Logs", command=top_logs, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_logs.pack(side=tk.LEFT, padx=10)

    button_fish= tk.Button(button_frame1, text="Fish", command=top_fish, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_fish.pack(side=tk.LEFT, padx=10)

    #panel 2-button frame 2 dia ta katw 4 koumpia
    button_frame2 = tk.Frame(new_window, bg="blue")
    button_frame2.pack(side='bottom', fill='both', expand=True, padx=10, pady=10)

    button_fruit= tk.Button(button_frame2, text="Fruit", command=top_fruit, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_fruit.pack(side=tk.LEFT, padx=10)

    button_nonfood= tk.Button(button_frame2, text="NonFood", command=top_non_food, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_nonfood.pack(side=tk.LEFT, padx=10)

    button_mechanical= tk.Button(button_frame2, text="Mechanical", command=top_mechanical, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_mechanical.pack(side=tk.LEFT, padx=10)

    button_electrical= tk.Button(button_frame2, text="Electrical", command=top_electrical, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_electrical.pack(side=tk.LEFT, padx=10)

```

```

"""HOME PAGE WITH BUTTONS"""
#ftiaxnw to main frame me ta buttons
root = tk.Tk()
root.title("Covid Effects on Trades")

#main frame
main_frame = tk.Frame(root, bg="blue")
main_frame.pack(fill='both', expand=True)

#subframe1 dia ta panw buttons, afirw kena endiamesa
button_frame1 = tk.Frame(main_frame, bg="blue")
button_frame1.pack(side='top', fill='both', expand=True, padx=10, pady=10)

#buttons gia subframe1
button_month = tk.Button(button_frame1, text="Total Value per Month", command=graph_value_per_month, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
button_month.pack(side=tk.LEFT, padx=10)

button_country = tk.Button(button_frame1, text="Total Value per Country", command=graph_value_per_country, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
button_country.pack(side=tk.LEFT, padx=10)

button_transport= tk.Button(button_frame1, text="Total Value per Transport", command=graph_value_per_transport, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
button_transport.pack(side=tk.LEFT, padx=10)

#subframe2 dia ta katw buttons, afirw kena endiamesa
button_frame2 = tk.Frame(main_frame, bg="blue")
button_frame2.pack(side='bottom', fill='both', expand=True, padx=10, pady=10)

#buttons gia subframe2
button_day= tk.Button(button_frame2, text="Total Value per Day", command=graph_value_per_day, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
button_day.pack(side=tk.LEFT, padx=10)

button_commodity= tk.Button(button_frame2, text="Total Value per Commodity", command=graph_value_per_commodity, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
button_commodity.pack(side=tk.LEFT, padx=10)

button_top_5= tk.Button(button_frame2, text="Top 5 valued months", command=graph_5_months, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
button_top_5.pack(side=tk.LEFT, padx=10)

#subframe3 dia ta endiamesa buttons, afirw kena endiamesa
button_frame3 = tk.Frame(main_frame, bg="blue")
button_frame3.pack(side='bottom', fill='both', expand=True, padx=10, pady=10)

#anoiwg nea frame me ta buttons=countries
button_top_5_commodities_per_country=tk.Button(button_frame3, text="Commodities/Country", command=new_window_for_commodities, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
button_top_5_commodities_per_country.pack(side=tk.LEFT, padx=10)

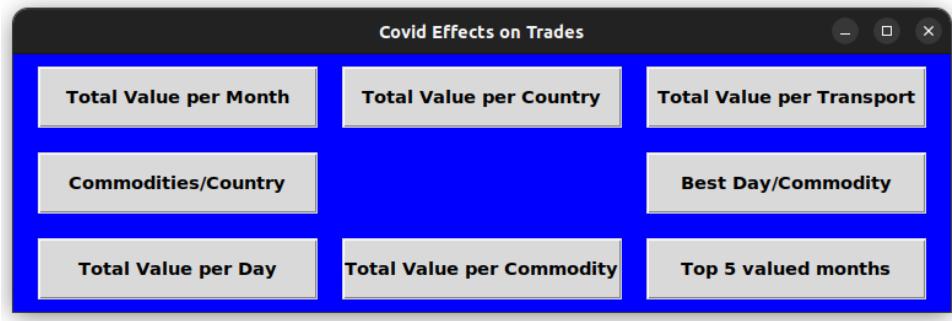
#anoiwg nea frame me ta buttons=commodities
button_top_day_per_commodity=tk.Button(button_frame3, text="Best Day/Commodity", command=new_window_for_days, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
button_top_day_per_commodity.pack(side=tk.RIGHT, padx=10)

#treixi main loop
root.mainloop()

#kleirw to connection me to db
cnx.close()

```

To GUI που εμφανίζεται είναι το εξής:

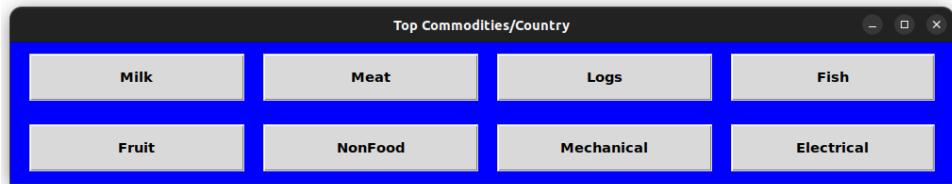


Πατώντας κάποιο κουμπί εμφανίζεται το εκάστοτε γράφημα.

Αν πατήσει κάποιος το Commodities/Country εμφανίζεται αυτό το frame όπου ο χρήστης μπορεί να επιλέξει τη χώρα για την οποία θέλει να δει το γράφημα.



Αντίστοιχα, αν κάποιος επιλέξει το Best Day/Commodity εμφανίζεται αυτό το frame όπου ο χρήστης μπορεί να επιλέξει την κατηγορία εμπορεύματος για την οποία θέλει να δει το γράφημα.



Επίσης, εμφανίζονται κατάλληλα μηνύματα για το τι πρόκειται να δει ο χρήστης αν πατήσει κάποιο κουμπί.



## **E) Βιβλιογραφία:**

- Διαφάνειες κ. Βονιτσάνου  
*(Eclass - μάθημα: Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών)*
- Πραγματολογία των Γλωσσών Προγραμματισμού  
(Michael L. Scott, 2η Έκδοση, 2009, ΚΛΕΙΔΑΡΙΘΜΟΣ)

## F) Συνολικός Τελικός Κώδικας:

```
import requests
import csv
import matplotlib.pyplot as plt
import tkinter as tk
from tkinter import messagebox
import mysql.connector

url = "https://www.stats.govt.nz/assets/Uploads/Effects-of-COVID-19-on-trade/Effects-of-COVID-19-on-trade-At-15-December-2021-provisional/Download-data/effects-of-covid-19-on-trade-at-15-december-2021-provisional.csv"
counter=0

#sindesi sto database
cnx = mysql.connector.connect(user='root',
password='Melenegiorgo2002_', host='localhost', database='Covid_Effects')

#dictionaries
indexValue_per_month = {}
indexValue_per_country={}
indexValue_per_transport={}
indexValue_per_day={}
indexValue_per_commodity={}
indexCommodities_per_country = {}

#dictionaries gia top 5 commodities/country
indexChina={}
indexAustralia={} #exei mono ALL commodity gia auto den to vazw
```

```
indexUnitedStates={}
indexJapan={}
indexUnitedKingdom={}
indexEuropeanUnion={}
indexEastAsia={}
```

#indexes gia top day/commodity

```
milk = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
```

```
meat = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
```

```
logs = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
```

```
fish = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
```

```
fruit = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
```

```
non_food = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
```

```
mechanical = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
```

```
electrical = {'Monday': 0, 'Tuesday': 0, 'Wednesday': 0, 'Thursday': 0, 'Friday': 0, 'Saturday': 0, 'Sunday': 0}
```

with requests.Session() as s:

```
#download=url
download = s.get(url)
```

"""\edw diavazw kai typwnw to .csv"""

```
if download.status_code == 200:  
    data = download.text  
    csv_data = csv.reader(data.splitlines())  
    for row in csv_data:  
        print(row)  
  
else:  
    print("Error: Failed to download CSV data.")  
  
"""anoigma tou csv gia epeksergasia"""  
decoded_content = download.content.decode('utf-8')  
  
data = csv.reader(decoded_content.splitlines(), delimiter=',')  
  
#pernaw tin prwti grammi pou exei ta onomata stilwn  
next(data)  
  
#vazw ta data se ena list wste na mporw na ta xrisimopoihsw kai meta  
to 1o iteration  
data_list = list(data)  
value=float(row[8])  
date=row[2]  
day = date.split("/")[0]  
  
"""kwdkas gia tziro/month"""  
#pernaw kathe grammi tou .csv kai upologizw to value  
for row in data_list:  
    date = row[2] #date in the third column
```

```
month = date.split("//")[1] #pernw ton mina apo to date sto pos1 tou  
date
```

```
if month in indexValue_per_month:  
    indexValue_per_month[month] += value  
else:  
    indexValue_per_month[month] = value
```

"""\kwdikas gia tziro/country"""

```
for row in data_list:  
    country = row[4] #country in the fifth column
```

```
#xwris to "All" kai to Total  
if country!="All" and country!="Total (excluding China)":  
    if country in indexValue_per_country:  
        indexValue_per_country[country] += value  
    else:  
        indexValue_per_country[country] = value
```

"""\kwdikas gia tziro/transport"""

```
for row in data_list:  
    transport_mode=row[6]  
  
    #xwris to "All"  
    if transport_mode!="All":  
        if transport_mode in indexValue_per_transport:  
            indexValue_per_transport[transport_mode]+=value
```

```

else:
    indexValue_per_transport[transport_mode]=value

"""
kwdikas gia tziro/day"""

#pernw kai the grammi tou .csv kai upologizw to value
for row in data_list:
    date = row[2] #date in the third column
    day = date.split("/")[0] #pernw ti mera apo to date, sto pos0 tou date

    if day in indexValue_per_day:
        indexValue_per_day[day] += value
    else:
        indexValue_per_day[day] = value

"""
kwdikas gia tziro/commodity"""

for row in data_list:
    commodity=row[5]

    #xwrис to "All"
    if commodity!="All":
        if commodity in indexValue_per_commodity:
            indexValue_per_commodity[commodity]+=value
        else:
            indexValue_per_commodity[commodity]=value

"""
kwdikas for top 5 valued commodities per country CHINA"""

for row in data_list:

```

```

commodity=row[5]
country=row[4]

if commodity!="All" and country=="China":
    if commodity in indexChina:
        indexChina[commodity]+=value
    else:
        indexChina[commodity]=value

"""kwedikas for top 5 valued commodities per country UNITED STATES"""
for row in data_list:

    commodity=row[5]
    country=row[4]

    if commodity!="All" and country=="United States":
        if commodity in indexUnitedStates:
            indexUnitedStates[commodity]+=value
        else:
            indexUnitedStates[commodity]=value

"""kwedikas for top 5 valued commodities per country ASIA"""
for row in data_list:
    commodity=row[5]
    country=row[4]
    if commodity!="All" and country=="East Asia (excluding China)":
        if commodity in indexEastAsia:
            indexEastAsia[commodity]+=value

```

```
else:  
    indexEastAsia[commodity]=value  
  
    """top days for milk"""  
    for row in data_list:  
        commodity=row[5]  
        weekday=row[3]  
  
        if commodity=="Milk powder, butter, and cheese":  
            if weekday in milk:  
                milk[weekday]+=value  
            else:  
                milk[weekday]=value  
  
    """top days for meat"""  
    for row in data_list:  
        commodity=row[5]  
        weekday=row[3]  
  
        if commodity=="Meat and edible offal":  
            if weekday in milk:  
                meat[weekday]+=value  
            else:  
                meat[weekday]=value  
  
    """top days for logs"""  
    for row in data_list:  
        commodity=row[5]  
        weekday=row[3]
```

```
if commodity=="Logs, wood, and wood articles":  
    if weekday in logs:  
        logs[weekday]+=value  
    else:  
        logs[weekday]=value  
  
"""top days for fish"""  
for row in data_list:  
    commodity=row[5]  
    weekday=row[3]  
  
    if commodity=="Fish, crustaceans, and molluscs":  
        if weekday in fish:  
            fish[weekday]+=value  
        else:  
            fish[weekday]=value  
  
"""top days for fruit"""  
for row in data_list:  
    commodity=row[5]  
    weekday=row[3]  
  
    if commodity=="Fruit":  
        if weekday in fruit:  
            fruit[weekday]+=value  
        else:  
            fruit[weekday]=value
```

```
"""top days for non food"""
for row in data_list:
    commodity=row[5]
    weekday=row[3]

    if commodity=="Non-food manufactured goods":
        if weekday in fruit:
            non_food[weekday]+=value
        else:
            non_food[weekday]=value

"""top days for mechanical"""
for row in data_list:
    commodity=row[5]
    weekday=row[3]

    if commodity=="Mechanical machinery and equip":
        if weekday in fruit:
            mechanical[weekday]+=value
        else:
            mechanical[weekday]=value

"""top days for electrical"""
for row in data_list:
    commodity=row[5]
    weekday=row[3]

    if commodity=="Electrical machinery and equip":
        if weekday in fruit:
```

```

electrical[weekday]+=value
else:
    electrical[weekday]=value

"""
graph gia tziro/month"""

def graph_value_per_month():

    messagebox.showinfo("Message", "You will see the value/month
graph.")

    #vriskw ti megisti timi
    max_value = maxValue_per_month.values()

    #vriskw ta transport pou exoun idio max value
    #kanei iterate sta key-value pairs tou indexValuePerMonth kai elegxei
    #an to value einai equal me max_months
    #telika krataei megalitero
    max_months = [k for k, v in maxValue_per_month.items() if v ==
max_value]

    #plots
    fig, ax = plt.subplots(figsize=(10,6))
    bars = ax.bar(indexValue_per_month.keys(),
indexValue_per_month.values())

    #idio xrwma sta max values
    for month in max_months:
        bars[list(indexValue_per_month.keys()).index(month)].set_color('magenta')

```

```

#legend

blue_patch = plt.Rectangle((0, 0), 1, 1, color='blue')
magenta_patch = plt.Rectangle((0, 0), 1, 1, color='magenta')
plt.legend([blue_patch, magenta_patch], ['Value', 'Max Value'],
loc='upper right')

#first graph: tziros/month

plt.title("Total Value per Month")
plt.xlabel("Month")
plt.ylabel("Value (in dollars)")
plt.show()

#gia sql

cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS ValuePerMonth")
cursor.execute("CREATE TABLE ValuePerMonth (month INT, value
BIGINT)")

for month, indexValue in indexValue_per_month.items():

    cursor.execute("INSERT INTO ValuePerMonth VALUES (%s, %s)",
(month, indexValue))

cnx.commit()

#gia .csv files

with open('ValuePerMonth.csv', 'w', newline='') as csvfile:

    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM ValuePerMonth")

    #gia ta onoma twn stilwn
    csvwriter.writerow([i[0] for i in cursor.description])

```

```

csvwriter.writerows(cursor)

cursor.close()

"""graph gia tziro/country"""
def graph_value_per_country():

    messagebox.showinfo("Message", "You will see the value/country
graph.")

    #vriskw ti megisti timi
    max_value = max(indexValue_per_country.values())

    #vriskw ta transport pou exoun idio max value
    max_countries = [k for k, v in indexValue_per_country.items() if v ==
max_value]

    #plots
    fig, ax = plt.subplots(figsize=(18,6))

    bars = ax.bar(indexValue_per_country.keys(),
indexValue_per_country.values())

    #idio xrwma sta max values
    for country in max_countries:

        bars[list(indexValue_per_country.keys()).index(country)].set_color('magenta')

    #legend
    blue_patch = plt.Rectangle((0, 0), 1, 1, color='blue')
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='magenta')

```

```
    plt.legend([blue_patch, magenta_patch], ['Value', 'Max Value'],  
loc='upper right')
```

```
#second graph: tziros/country  
plt.title("Total Value per Country")  
plt.xlabel("Country")  
plt.ylabel("Value (in dollars)")  
plt.show()
```

```
#gia sql  
cursor=cnx.cursor()  
cursor.execute("DROP TABLE IF EXISTS ValuePerCountry")  
cursor.execute("CREATE TABLE ValuePerCountry (country  
VARCHAR(255), value BIGINT)")
```

```
for country, indexValue in indexValue_per_country.items():  
    cursor.execute("INSERT INTO ValuePerCountry VALUES (%s,  
%s)", (country, indexValue))
```

```
cnx.commit()
```

```
#gia .csv file  
with open('ValuePerCountry.csv', 'w', newline='') as csvfile:  
    csvwriter = csv.writer(csvfile)  
    cursor.execute("SELECT * FROM ValuePerCountry")  
    csvwriter.writerow([i[0] for i in cursor.description])  
    csvwriter.writerows(cursor)
```

```
cursor.close()
```

```

"""graph gia tziro/transport"""
def graph_value_per_transport():

    messagebox.showinfo("Message", "You will see the
value/transport_mode graph.")

    #vriskw ti megisti timi
    max_value = maxValue_per_transport.values()

    #vriskw ta transport pou exoun idio max value
    max_transports = [k for k, v in maxValue_per_transport.items() if v
== max_value]

    #plots
    fig, ax = plt.subplots(figsize=(10,6))

    bars = ax.bar(indexValue_per_transport.keys(),
indexValue_per_transport.values())

    #idio xrwma sta max values
    for transport in max_transports:

        bars[list(indexValue_per_transport.keys()).index(transport)].set_color('magenta')

    #legend
    blue_patch = plt.Rectangle((0, 0), 1, 1, color='blue')
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='magenta')
    plt.legend([blue_patch, magenta_patch], ['Value', 'Max Value'],
loc='upper right')

    #third graph: tziros/transport
    plt.title("Total Value per Transport")

```

```

plt.xlabel("Transport")
plt.ylabel("Value (in dollars)")
plt.show()

#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS ValuePerTransport")
cursor.execute("CREATE TABLE ValuePerTransport (transport
VARCHAR(255), value BIGINT)")

for transport, indexValue in indexValue_per_transport.items():
    cursor.execute("INSERT INTO ValuePerTransport VALUES (%s,
%s)", (transport, indexValue))

cnx.commit()

#gia .csv file
with open('ValuePerTransport.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM ValuePerTransport")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()

"""graph gia tziro/day"""
def graph_value_per_day():

    messagebox.showinfo("Message", "You will see the value/day graph.")

```

```

#vriskw ti megisti timi
max_value = max(indexValue_per_day.values())

#vriskw ta days pou exoun idio max value
max_days = [k for k, v in indexValue_per_day.items() if v == max_value]

#plots
fig, ax = plt.subplots(figsize=(12,6))
bars = ax.bar(indexValue_per_day.keys(),
indexValue_per_day.values())

#idio xrwma sta max values
for day in max_days:
    bars[list(indexValue_per_day.keys()).index(day)].set_color('magenta')

#legend
blue_patch = plt.Rectangle((0, 0), 1, 1, color='blue')
magenta_patch = plt.Rectangle((0, 0), 1, 1, color='magenta')
plt.legend([blue_patch, magenta_patch], ['Value', 'Max Value'],
loc='upper right')

#forth graph: tziros/transport
plt.title("Total Value per Day")
plt.xlabel("Day")
plt.ylabel("Value (in dollars)")
plt.show()

```

```

#gia sql

cursor=cnx.cursor()

cursor.execute("DROP TABLE IF EXISTS ValuePerDay")

cursor.execute("CREATE TABLE ValuePerDay (day INT, value
BIGINT)")

for day, indexValue in indexValue_per_day.items():

    cursor.execute("INSERT INTO ValuePerDay VALUES (%s, %s)",
(day, indexValue))

cnx.commit()

#gia .csv file

with open('ValuePerDay.csv', 'w', newline='') as csvfile:

    csvwriter = csv.writer(csvfile)

    cursor.execute("SELECT * FROM ValuePerDay")

    csvwriter.writerow([i[0] for i in cursor.description])

    csvwriter.writerows(cursor)

cursor.close()

"""graph gia tziro/commodity"""

def graph_value_per_commodity():

    messagebox.showinfo("Message", "You will see the value/commodity
graph.")

    #vriskw ti megisti timi

    max_value = max(indexValue_per_commodity.values())

    #vriskw ta commodities pou exoun idio max value

```

```
max_commodities = [k for k, v in indexValue_per_commodity.items() if
v == max_value]
```

```
#plots
fig, ax = plt.subplots(figsize=(16,10))
bars = ax.bar(indexValue_per_commodity.keys(),
indexValue_per_commodity.values())
```

```
#idio xrwma sta max values
for commodity in max_commodities:
```

```
bars[list(indexValue_per_commodity.keys()).index(commodity)].set_color('mag
enta')
```

```
#vazw klisi sta bar names gia na xwrane
plt.xticks(rotation=20)
```

```
#legend
blue_patch = plt.Rectangle((0, 0), 1, 1, color='blue')
magenta_patch = plt.Rectangle((0, 0), 1, 1, color='magenta')
plt.legend([blue_patch, magenta_patch], ['Value', 'Max Value'],
loc='upper right')
```

```
#fifth graph: tziros/transport
plt.title("Total Value per Commodity")
plt.xlabel("Commodity")
plt.ylabel("Value (in dollars)")
plt.show()
```

```
#gia sql
```

```

cursor=cnx.cursor()

cursor.execute("DROP TABLE IF EXISTS ValuePerCommodity")

cursor.execute("CREATE TABLE ValuePerCommodity (commodity
VARCHAR(255), value BIGINT)")

for commodity, indexValue in indexValue_per_transport.items():

    cursor.execute("INSERT INTO ValuePerCommodity VALUES (%s,
%s)", (commodity, indexValue))

cnx.commit()

#gia .csv file

with open('ValuePerCommodity.csv', 'w', newline='') as csvfile:

    csvwriter = csv.writer(csvfile)

    cursor.execute("SELECT * FROM ValuePerCommodity")

    csvwriter.writerow([i[0] for i in cursor.description])

    csvwriter.writerows(cursor)

cursor.close()

"""graph gia 5 mines me megalitero tziro"""

def graph_5_months():

    messagebox.showinfo("Message", "You will see the 5 most valued
months graph.")

    #kanw sort to index gia na parw meta tous megaliterous orous

    #lambda giati apo ta turples pou epistrefontai kratame to deutero item
    x[1], dld to value

```

```

sorted_months=sorted(indexValue_per_month.items(), key=lambda x:
x[1], reverse=True)

#pairnw tis 5 times
top_5_months=sorted_months[:5]

#plot
fig, ax = plt.subplots(figsize=(10, 6))
ax.bar([x[0] for x in top_5_months], [x[1] for x in top_5_months],
color='magenta')

#legend
blue_patch = plt.Rectangle((0, 0), 1, 1, color='blue')
magenta_patch = plt.Rectangle((0, 0), 1, 1, color='magenta')
plt.legend([blue_patch, magenta_patch], ['Value', 'Max Value'],
loc='upper right')

#sixth graph
ax.set_xlabel('Month')
ax.set_ylabel('Value (in dollars)')
ax.set_title('Most valued months')
plt.show()

#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopFiveMonths")
cursor.execute("CREATE TABLE TopFiveMonths (month INT, value
BIGINT)")

for month, indexValue in top_5_months:
    cursor.execute("INSERT INTO TopFiveMonths (month, value) VALUES (%s, %s)", (month, indexValue))

cnx.commit()
cursor.close()

```

```
        cursor.execute("INSERT INTO TopFiveMonths VALUES (%s, %s)",  
(month, indexValue))
```

```
    cnx.commit()
```

```
#gia .csv file  
with open('TopFiveMonths.csv', 'w', newline='') as csvfile:  
    csvwriter = csv.writer(csvfile)  
    cursor.execute("SELECT * FROM TopFiveMonths")  
    csvwriter.writerow([i[0] for i in cursor.description])  
    csvwriter.writerows(cursor)
```

```
cursor.close()
```

```
"""graphs gia top 5 valued commodities/country"""
```

```
def graph_5_commodities_china():
```

```
    messagebox.showinfo("Message", "You will see the graph about the 5  
most valued commodities for china.")
```

```
    sorted_commodities_china=sorted(indexChina.items(), key=lambda x:  
x[1], reverse=True)
```

```
    top_5_commodities=sorted_commodities_china[:5]
```

```
    fig, ax = plt.subplots(figsize=(18, 6))
```

```
    ax.bar([x[0] for x in top_5_commodities], [x[1] for x in  
top_5_commodities], color='red')
```

```
#legend
```

```
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
```

```
plt.legend([magenta_patch], ['Value'], loc='upper right')

ax.set_xlabel('Commodities')
ax.set_ylabel('China')
ax.set_title('Most valued commodities')
plt.show()

#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopCommoditiesChina")
cursor.execute("CREATE TABLE TopCommoditiesChina (commodity
VARCHAR(255), value BIGINT)")

for commodity, indexPath in top_5_commodities:
    cursor.execute("INSERT INTO TopCommoditiesChina VALUES (%s,
%s)", (commodity, indexPath))

cnx.commit()

#gia .csv file
with open('TopCommoditiesChina.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopCommoditiesChina")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()

def graph_5_commodities_US():
```

```
messagebox.showinfo("Message", "You will see the graph about the 5  
most valued commodities for US.")
```

```
sorted_commodities_US=sorted(indexUnitedStates.items(),  
key=lambda x: x[1], reverse=True)
```

```
top_5_commodities=sorted_commodities_US[:5]
```

```
fig, ax = plt.subplots(figsize=(4, 6))  
  
ax.bar([x[0] for x in top_5_commodities], [x[1] for x in  
top_5_commodities], color='red')
```

```
#legend
```

```
magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')  
plt.legend([magenta_patch], ['Value'], loc='upper right')
```

```
ax.set_xlabel('Commodities')  
ax.set_ylabel('United States')  
ax.set_title('Most valued commodities')  
plt.show()
```

```
#gia sql
```

```
cursor=cnx.cursor()  
cursor.execute("DROP TABLE IF EXISTS TopCommoditiesUS")  
cursor.execute("CREATE TABLE TopCommoditiesUS (commodity  
VARCHAR(255), value BIGINT)")
```

```
for commodity, indexValue in top_5_commodities:
```

```
cursor.execute("INSERT INTO TopCommoditiesUS VALUES (%s,  
%s)", (commodity, indexValue))
```

```
cnx.commit()
```

```
#gia .csv file
```

```
with open('TopCommoditiesUS.csv', 'w', newline=') as csvfile:
```

```
    csvwriter = csv.writer(csvfile)
```

```
    cursor.execute("SELECT * FROM TopCommoditiesUS")
```

```
    csvwriter.writerow([i[0] for i in cursor.description])
```

```
    csvwriter.writerows(cursor)
```

```
cursor.close()
```

```
def graph_5_commodities_EA():
```

```
    messagebox.showinfo("Message", "You will see the graph about the 5  
most valued commodities for EA.")
```

```
    sorted_commodities_ea=sorted(indexEastAsia.items(), key=lambda x:  
x[1], reverse=True)
```

```
    top_5_commodities=sorted_commodities_ea[:5]
```

```
    fig, ax = plt.subplots(figsize=(4, 6))
```

```
    ax.bar([x[0] for x in top_5_commodities], [x[1] for x in  
top_5_commodities], color='red')
```

```
#legend
```

```
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
```

```
    plt.legend([magenta_patch], ['Value'], loc='upper right')
```

```

    ax.set_xlabel('Commodities')
    ax.set_ylabel('East Asia (excluding china)')
    ax.set_title('Most valued commodities')
    plt.show()

#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopCommoditiesEA")
cursor.execute("CREATE TABLE TopCommoditiesEA (commodity
VARCHAR(255), value BIGINT)")

for commodity, indexValue in top_5_commodities:
    cursor.execute("INSERT INTO TopCommoditiesEA VALUES (%s,
%s)", (commodity, indexValue))

cnx.commit()

#gia .csv file
with open('TopCommoditiesEA.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopCommoditiesEA")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()

"""graphs gia top day/commodity"""
def top_milk():

```

```
messagebox.showinfo("Message", "You will see the graph about the  
most valued day for milk powder, butter, and cheese.")
```

```
sorted_days=sorted(milk.items(), key=lambda x: x[1], reverse=True)  
top5=sorted_days[:1]
```

```
fig, ax = plt.subplots(figsize=(5, 6))  
ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')
```

```
#legend  
magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')  
plt.legend([magenta_patch], ['Value'], loc='upper right')
```

```
ax.set_xlabel('Day')  
ax.set_ylabel('Milk powder, butter, and cheese')  
ax.set_title('Most valued day')  
plt.show()
```

```
#gia sql  
cursor=cnx.cursor()  
cursor.execute("DROP TABLE IF EXISTS TopDayForMilk")  
cursor.execute("CREATE TABLE TopDayForMilk (commodity  
VARCHAR(255), value BIGINT)")
```

```
for commodity, indexValue in top5:  
    cursor.execute("INSERT INTO TopDayForMilk VALUES (%s, %s)",  
(commodity, indexValue))
```

```
cnx.commit()
```

```

#gia .csv file

with open('TopDayForMilk.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopDayForMilk")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()

def top_meat():

    messagebox.showinfo("Message", "You will see the graph about the
most valued day for meat and edible offal")

    sorted_days=sorted(meat.items(), key=lambda x: x[1], reverse=True)
    top5=sorted_days[:1]

    fig, ax = plt.subplots(figsize=(5, 6))
    ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')

    #legend
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
    plt.legend([magenta_patch], ['Value'], loc='upper right')

    ax.set_xlabel('Day')
    ax.set_ylabel('Meat and edible offal')
    ax.set_title('Most valued day')
    plt.show()

```

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopDayForMeat")
cursor.execute("CREATE TABLE TopDayForMeat (commodity
VARCHAR(255), value BIGINT)")
```

*for commodity, indexPath in top5:*

```
cursor.execute("INSERT INTO TopDayForMeat VALUES (%s, %s)",
(commodity, indexPath))
```

```
cnx.commit()
```

*#gia .csv file*

*with open('TopDayForMeat.csv', 'w', newline='') as csvfile:*

```
csvwriter = csv.writer(csvfile)
cursor.execute("SELECT * FROM TopDayForMeat")
csvwriter.writerow([i[0] for i in cursor.description])
csvwriter.writerows(cursor)
```

```
cursor.close()
```

*def top\_logs():*

```
messagebox.showinfo("Message", "You will see the graph about the
most valued day for logs, wood, and wood articles")
```

```
sorted_days=sorted(logs.items(), key=lambda x: x[1], reverse=True)
top5=sorted_days[:1]
```

```
fig, ax = plt.subplots(figsize=(5, 6))
ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')
```

```
#legend
magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
plt.legend([magenta_patch], ['Value'], loc='upper right')
```

```
ax.set_xlabel('Day')
ax.set_ylabel('Logs, wood, and wood articles')
ax.set_title('Most valued day')
plt.show()
```

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopDayForWood")
cursor.execute("CREATE TABLE TopDayForWood (commodity
VARCHAR(255), value BIGINT)")
```

*for commodity, indexValue in top5:*

```
cursor.execute("INSERT INTO TopDayForWood VALUES (%s,
%s)", (commodity, indexValue))
```

```
cnx.commit()
```

```
#gia .csv file
with open('TopDayForWood.csv', 'w', newline='') as csvfile:
csvwriter = csv.writer(csvfile)
cursor.execute("SELECT * FROM TopDayForWood")
csvwriter.writerow([i[0] for i in cursor.description])
```

```
csvwriter.writerows(cursor)

cursor.close()

def top_fish():

    messagebox.showinfo("Message", "You will see the graph about the
most valued day for fish, crustaceans, and molluscs")

    sorted_days=sorted(fish.items(), key=lambda x: x[1], reverse=True)
    top5=sorted_days[:1]

    fig, ax = plt.subplots(figsize=(5, 6))
    ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')

    #legend
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
    plt.legend([magenta_patch], ['Value'], loc='upper right')

    ax.set_xlabel('Day')
    ax.set_ylabel('Fish, crustaceans, and molluscs')
    ax.set_title('Most valued day')
    plt.show()

#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopDayForFish")
cursor.execute("CREATE TABLE TopDayForFish (commodity
VARCHAR(255), value BIGINT)")
```

```

for commodity, indexValue in top5:
    cursor.execute("INSERT INTO TopDayForFish VALUES (%s, %s)",
    (commodity, indexValue))

cnx.commit()

#gia .csv file
with open('TopDayForFish.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopDayForFish")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()

def top_fruit():

    messagebox.showinfo("Message", "You will see the graph about the
most valued day for fruit")

    sorted_days=sorted(fruit.items(), key=lambda x: x[1], reverse=True)
    top5=sorted_days[:1]

    fig, ax = plt.subplots(figsize=(5, 6))
    ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')

#legend
magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
plt.legend([magenta_patch], ['Value'], loc='upper right')

```

```

    ax.set_xlabel('Day')
    ax.set_ylabel('Fruit')
    ax.set_title('Most valued day')
    plt.show()

#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopDayForFruit")
cursor.execute("CREATE TABLE TopDayForFruit (commodity
VARCHAR(255), value BIGINT)")

for commodity, indexValue in top5:
    cursor.execute("INSERT INTO TopDayForFruit VALUES (%s, %s)",
(commodity, indexValue))

cnx.commit()

#gia .csv file
with open('TopDayForFruit.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopDayForFruit")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)

cursor.close()

def top_non_food():
    messagebox.showinfo("Message", "You will see the graph about the
most valued day for non-food manufactured goods")

```

```
sorted_days=sorted(non_food.items(),      key=lambda      x:      x[1],  
reverse=True)
```

```
top5=sorted_days[:1]
```

```
fig, ax = plt.subplots(figsize=(5, 6))
```

```
ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')
```

```
#legend
```

```
magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
```

```
plt.legend([magenta_patch], ['Value'], loc='upper right')
```

```
ax.set_xlabel('Day')
```

```
ax.set_ylabel('Non-food manufactured goods')
```

```
ax.set_title('Most valued day')
```

```
plt.show()
```

```
#gia sql
```

```
cursor=cnx.cursor()
```

```
cursor.execute("DROP TABLE IF EXISTS TopDayForNonFood")
```

```
cursor.execute("CREATE TABLE TopDayForNonFood (commodity  
VARCHAR(255), value BIGINT)")
```

```
for commodity, indexValue in top5:
```

```
    cursor.execute("INSERT INTO TopDayForNonFood VALUES (%s,  
%s)", (commodity, indexValue))
```

```
cnx.commit()
```

```
#gia .csv file
```

```
with open('TopDayForNonFood.csv', 'w', newline='') as csvfile:  
    csvwriter = csv.writer(csvfile)  
    cursor.execute("SELECT * FROM TopDayForNonFood")  
    csvwriter.writerow([i[0] for i in cursor.description])  
    csvwriter.writerows(cursor)  
  
cursor.close()  
  
def top_mechanical():  
  
    messagebox.showinfo("Message", "You will see the graph about the  
most valued day for mechanical machinery and equip")  
  
    sorted_days=sorted(mechanical.items(), key=lambda x: x[1],  
reverse=True)  
    top5=sorted_days[:1]  
  
    fig, ax = plt.subplots(figsize=(5, 6))  
    ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')  
  
    #legend  
    magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')  
    plt.legend([magenta_patch], ['Value'], loc='upper right')  
  
    ax.set_xlabel('Day')  
    ax.set_ylabel('Mechanical machinery and equip')  
    ax.set_title('Most valued day')  
    plt.show()
```

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopDayMechanical")
cursor.execute("CREATE TABLE TopDayMechanical (commodity
VARCHAR(255), value BIGINT)")
```

*for commodity, indexPath in top5:*

```
    cursor.execute("INSERT INTO TopDayMechanical VALUES (%s,
%s)", (commodity, indexPath))
```

```
cnx.commit()
```

*#gia .csv file*

*with open('TopDayMechanical.csv', 'w', newline='') as csvfile:*

```
    csvwriter = csv.writer(csvfile)
    cursor.execute("SELECT * FROM TopDayMechanical")
    csvwriter.writerow([i[0] for i in cursor.description])
    csvwriter.writerows(cursor)
```

```
cursor.close()
```

*def top\_electrical():*

*messagebox.showinfo("Message", "You will see the graph about the  
most valued day for electrical machinery and equip")*

```
    sorted_days=sorted(electrical.items(),      key=lambda      x:      x[1],
reverse=True)
```

```
    top5=sorted_days[:1]
```

```
fig, ax = plt.subplots(figsize=(5, 6))
ax.bar([x[0] for x in top5], [x[1] for x in top5], color='red')
```

```
#legend
magenta_patch = plt.Rectangle((0, 0), 1, 1, color='red')
plt.legend([magenta_patch], ['Value'], loc='upper right')
```

```
ax.set_xlabel('Day')
ax.set_ylabel('Electrical machinery and equip')
ax.set_title('Most valued day')
plt.show()
```

```
#gia sql
cursor=cnx.cursor()
cursor.execute("DROP TABLE IF EXISTS TopDayElectrical")
cursor.execute("CREATE TABLE TopDayElectrical (commodity
VARCHAR(255), value BIGINT)")
```

*for commodity, indexValue in top5:*

```
cursor.execute("INSERT INTO TopDayElectrical VALUES (%s,
%s)", (commodity, indexValue))
```

```
cnx.commit()
```

```
#gia .csv file
with open('TopDayElectrical.csv', 'w', newline='') as csvfile:
csvwriter = csv.writer(csvfile)
cursor.execute("SELECT * FROM TopDayElectrical")
csvwriter.writerow([i[0] for i in cursor.description])
```

```
csvwriter.writerows(cursor)
```

```
cursor.close()
```

```
"""gia na anoigw new window gia epilogi country poy psaxnw gia top 5  
valued commodities"""
```

```
def new_window_for_commodities():
```

```
    new_window = tk.Toplevel(root)
```

```
    new_window.title("Top Commodities/Country")
```

```
    new_window.configure(background='blue')
```

```
    new_window.geometry("740x200")
```

```
    button_china=      tk.Button(new_window,      text="China",  
command=graph_5_commodities_china, font=("TkDefaultFont", 11, "bold"),  
width=20, height=2)
```

```
    button_china.pack(side=tk.LEFT, padx=10)
```

```
    button_US=      tk.Button(new_window,      text="United States",  
command=graph_5_commodities_US, font=("TkDefaultFont", 11, "bold"),  
width=20, height=2)
```

```
    button_US.pack(side=tk.LEFT, padx=10)
```

```
    button_EA=      tk.Button(new_window,      text="East Asia",  
command=graph_5_commodities_EA, font=("TkDefaultFont", 11, "bold"),  
width=20, height=2)
```

```
    button_EA.pack(side=tk.LEFT, padx=10)
```

```
"""gia na anoigw new window gia epilogi commodity poy psaxnw gia top  
valued day"""
```

```

def new_window_for_days():

    new_window = tk.Toplevel(root)
    new_window.title("Top Commodities/Country")
    new_window.configure(background='blue')
    new_window.geometry("1000x150")

    #panel 1=button frame 1 gia ta panw 4 koumpia
    button_frame1 = tk.Frame(new_window, bg="blue")
    button_frame1.pack(side="top", fill="both", expand=True, padx=10,
pady=10)

    button_milk=          tk.Button(button_frame1,           text="Milk",
command=top_milk, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_milk.pack(side=tk.LEFT, padx=10)

    button_meat=          tk.Button(button_frame1,           text="Meat",
command=top_meat, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_meat.pack(side=tk.LEFT, padx=10)

    button_logs=          tk.Button(button_frame1,           text="Logs",
command=top_logs, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_logs.pack(side=tk.LEFT, padx=10)

    button_fish=          tk.Button(button_frame1,           text="Fish",
command=top_fish, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
    button_fish.pack(side=tk.LEFT, padx=10)

    #panel 2=button frame 2 gia ta katw 4 koumpia
    button_frame2 = tk.Frame(new_window, bg="blue")
    button_frame2.pack(side="bottom",   fill="both",   expand=True,
padx=10, pady=10)

```

```
button_fruit= tk.Button(button_frame2, text="Fruit",
command=top_fruit, font=("TkDefaultFont", 11, "bold"), width=20, height=2)
```

```
button_fruit.pack(side=tk.LEFT, padx=10)
```

```
button_nonfood= tk.Button(button_frame2, text="NonFood",
command=top_non_food, font=("TkDefaultFont", 11, "bold"), width=20,
height=2)
```

```
button_nonfood.pack(side=tk.LEFT, padx=10)
```

```
button_mechanical= tk.Button(button_frame2, text="Mechanical",
command=top_mechanical, font=("TkDefaultFont", 11, "bold"), width=20,
height=2)
```

```
button_mechanical.pack(side=tk.LEFT, padx=10)
```

```
button_electrical= tk.Button(button_frame2, text="Electrical",
command=top_electrical, font=("TkDefaultFont", 11, "bold"), width=20,
height=2)
```

```
button_electrical.pack(side=tk.LEFT, padx=10)
```

"""\bHOME PAGE WITH BUTTONS"""\b

#ftiaxnw to main frame me ta buttons

```
root = tk.Tk()
```

```
root.title("Covid Effects on Trades")
```

#main frame

```
main_frame = tk.Frame(root, bg="blue")
```

```
main_frame.pack(fill="both", expand=True)
```

```
#subframe1 gia ta panw buttons, afinw kena endiamesa
button_frame1 = tk.Frame(main_frame, bg="blue")
button_frame1.pack(side="top", fill="both", expand=True, padx=10,
pady=10)
```

*#buttons gia subframe1*

```
button_month = tk.Button(button_frame1, text="Total Value per Month",
command=graph_value_per_month, font=("TkDefaultFont", 11, "bold"),
width=20, height=2)
```

```
button_month.pack(side=tk.LEFT, padx=10)
```

```
button_country = tk.Button(button_frame1, text="Total Value per
Country", command=graph_value_per_country, font=("TkDefaultFont", 11,
"bold"), width=20, height=2)
```

```
button_country.pack(side=tk.LEFT, padx=10)
```

```
button_transport= tk.Button(button_frame1, text="Total Value per
Transport", command=graph_value_per_transport, font=("TkDefaultFont", 11,
"bold"), width=20, height=2)
```

```
button_transport.pack(side=tk.LEFT, padx=10)
```

*#subframe2 gia ta katw buttons, afinw kena endiamesa*

```
button_frame2 = tk.Frame(main_frame, bg="blue")
```

```
button_frame2.pack(side="bottom", fill="both", expand=True, padx=10,
pady=10)
```

*#buttons gia subframe2*

```
button_day= tk.Button(button_frame2, text="Total Value per Day",
command=graph_value_per_day, font=("TkDefaultFont", 11, "bold"),
width=20, height=2)
```

```
button_day.pack(side=tk.LEFT, padx=10)
```

```
button_commodity= tk.Button(button_frame2, text="Total Value per
Commodity", command=graph_value_per_commodity, font=("TkDefaultFont",
11, "bold"), width=20, height=2)
```

```
button_commodity.pack(side=tk.LEFT, padx=10)
```

```
button_top_5= tk.Button(button_frame2, text="Top 5 valued months",
command=graph_5_months, font=("TkDefaultFont", 11, "bold"), width=20,
height=2)
```

```
button_top_5.pack(side=tk.LEFT, padx=10)
```

```
#subframe3 gia ta endiamesa buttons, afinw kena endiamesa
```

```
button_frame3 = tk.Frame(main_frame, bg="blue")
```

```
button_frame3.pack(side="bottom", fill="both", expand=True, padx=10,
pady=10)
```

```
#anoigw neo frame me ta buttons=countries
```

```
button_top_5_commodities_per_countrry=tk.Button(button_frame3,
text="Commodities/Country", command=new_window_for_commodities,
font=("TkDefaultFont", 11, "bold"), width=20, height=2)
```

```
button_top_5_commodities_per_countrry.pack(side=tk.LEFT, padx=10)
```

```
#anoigw neo frame me ta buttons=commodities
```

```
button_top_day_per_commodity=tk.Button(button_frame3, text="Best
Day/Commodity", command=new_window_for_days, font=("TkDefaultFont",
11, "bold"), width=20, height=2)
```

```
button_top_day_per_commodity.pack(side=tk.RIGHT, padx=10)
```

*#trexw main loop*

*root.mainloop()*

*#kleinw to connectio me to db*

*cnx.close()*