Πανεπιστήμιο Πατρών

Τμήμα Μηχανικών Η/Υ
& πληροφορικής

Μάθημα: Αρχές Γλωσσών
Προγραμματισμού και Μεταφραστών

Python Project Έτος 2022-2023

Επώνυμο: Καραγιάννης
Όνομα: Γεώργιος
ΑΜ : 1084586

Περιεχόμενα :

Σελίδα

1.1 Παρουσίαση λειτουργικού PyCharm

Γιατί PyCharm?

Το PyCharm αποτελεί ένα ευρέως γνωστό εργαλείο σε προγραμματιστές από όλο τον κόσμο για την υλοποίηση και διαχείριση προγραμμάτων πάνω στην γλώσσα προγραμματισμού Python. Το συγκεκριμένο λειτουργικό παρέχει πολλές χρήσιμες λειτουργίες όπως η ανάγνωση σφαλμάτων, απασφαλμάτωση τους και οργάνωση του κώδικα μας σε ευανάγνωστα τμήματα. Επίσης η κονσόλα που παρέχεται στο λειτουργικό αυτό καθιστά το όλο περιβάλλον ευέλικτο στην γρήγορη και εύκολη εκτέλεση Python scripts, κάτι το οποίο είναι απαραίτητο όταν πρέπει να υλοποιηθούν project τέτοιου βάθους και σκέψης.

1.2 Παρουσίαση Βιβλιοθηκών

Οι Βιβλιοθήκες

- **Pandas:** Την συγκεκριμένη βιβλιοθήκη την χρησιμοποιήσαμε διότι μας παρέχει μια ευρεία γκάμα από εργαλεία που μπορούμε να τα χρησιμοποιήσουμε με ευκολία για την διαχείριση των δεδομένων του αρχείου μας. Μερικά από αυτά τα εργαλεία είναι: **pd.read_csv():**Αυτή η συνάρτηση χρησιμοποιήθηκε για την ανάγνωση του CSV αρχείου από ένα συγκεκριμένο URL και τη φόρτωση των δεδομένων σε ένα DataFrame: **df = pd.read_csv().** Επίσης ένα άλλο σημαντικό εργαλείο είναι η μέθοδος **groupby()** όπου την χρησιμοποιήσαμε προκειμένου να ομαδοποιήσουμε τις κατάλληλες στήλες που χρειαζόμασταν για τις γραφικές παραστάσεις μας.
- **Matplotlib:**Την συγκεκριμένη βιβλιοθήκη την χρησιμοποιήσαμε προκειμένου να καταφέρουμε να οπτικοποίησουμε στο βέλτιστό και χωρίς δυσκολία τα δεδομένα που συλλέξαμε από το CSV αρχείο μας. Μερικά από τα εργαλεία που μας βοήθησαν είναι : **Subplots()** με την λειτουργία των subplots καταφέραμε να εντάξουμε σε ένα παράθυρο δυο γραφήματα, πράγμα το οποίο συνέβαλε στην εξοικονόμηση χρόνου αλλά και στη σύγκριση των δεδομένων μεταξύ τους. Επίσης η μέθοδος **Pie Chart()** μας επιτρέπει να δημιουργήσουμε διαγράμματα πίτας για την καλύτερη αναπαράσταση των δεδομένων στον χρήστη της εφαρμογής.
- **MySql.Connector:** Η συγκεκριμένη βιβλιοθήκη μας βοήθησε στην ενσωμάτωση κώδικα SQL μέσα στο περιβάλλον της python, πράγμα το οποίο μας διευκόλυνε στην ενσωμάτωση των δεδομένων

από τις γραφικές παραστάσεις σε κατάλληλα Tables μέσα σε μια βάση δεδομένων που δημιουργήσαμε εμείς ονομαζόμενη: **Covid_19_Data**.

- **Tkinter:** Η βιβλιοθήκη αυτή αποτέλεσε το βασικό εργαλείο για την δημιουργία της γραφικής διεπαφής που μας ζητήθηκε, προκειμένου ο χρήστης να μπορεί να επεξεργαστεί και να αλληλοεπιδράσει με τα δεδομένα που προέκυψαν από την ενασχόληση με το CSV αρχείο.
- **CSV:** Η βιβλιοθήκη αυτή μας βοήθησε κατά την διάρκεια του project οπου έπρεπε να αποθηκεύσουμε σε κατάλληλο αρχείο CSV τα δεδομένα που θέσαμε στου πίνακες της βάσης μας με ευκολία μέσα από το περιβάλλον του PyCharm.

2ο Κεφάλαιο – Διαχείριση δεδομένων CSV

2.1 Μέθοδος groupby()

Η μέθοδος αύτη αποτέλεσε κάποιες από τις βασικές μας εντολές κατά την διάρκεια του project. Συγκεκριμένα η μέθοδος αυτή χρησιμοποιήθηκε προκειμένου να καταφέρουμε να ομαδοποιήσουμε και να συγκεντρώσουμε τα δεδομένα που χρειαζόμαστε για τις γραφικές μας παραστάσεις από το DataFrame που αποθηκεύσαμε το αρχείο CSV μας (**εντολή**: df = pd.read_csv(url)). Με την εντολή **df.groupby(['Country', 'Measure'])** καταφέρνουμε να ομαδοποιήσουμε τα δεδομένα του df βάσει των στηλών **Country** και **Measure** προκειμένου να φτιάξουμε την γραφική για τον συνολικό τζίρο κάθε χώρας, όπου τα δεδομένα για τον τζίρο βρίσκονται στην στήλη Measure και για τις χώρες στη στήλη Country. Την ίδια ομαδοποίηση κάναμε και για τα δεδομένα που θέλαμε και για τις άλλες γραφικές παραστάσεις Πχ:

- Για τον συνολικό τζίρο για κάθε Μήνα κάναμε: **df.groupby(['Month', 'Measure'])** όπου ομαδοποιούμε τα δεδομένα του df βάσει των στηλών **Date** και **Measure**.
- Για τον συνολικό τζίρο κάθε Μέσο Μεταφοράς κάναμε: **df.groupby(['Transport_Mode','Measure'])**όπου ομαδοποιούμε τα δεδομένα του df βάσει των στηλών '**Transport_Mode** και **Measure**.
- Για τον συνολικό τζίρο κάθε Μέρας της εβδομάδας κάναμε: **df.groupby(['Weekday', 'Measure'])** όπου ομαδοποιούμε τα δεδομένα του df βάσει των στηλών '**Weekday**'και **Measure**.

Και συνεχίζαμε κατά αυτόν τον τρόπο για όλες τις γραφικές μας παραστάσεις.

2.2 Μέθοδος sum()

Με την μέθοδο αυτή καταφέραμε να υπολογίσουμε το άθροισμά του συνολικού τζίρου από την στήλη Value για κάθε ομάδα που δημιουργήθηκε από την μέθοδο groupby().Έτσι για κάθε μοναδικό συνδυασμό των τιμών από την μέθοδο **df.groupby(['Country', 'Measure'])** η μέθοδος **sum** υπολογίζει το άθροισμα των αντίστοιχων τιμών στη στήλη 'Value'.

Η εντολή την οποία εκτελούμε είναι η ακόλουθη: **totals = grouped['Value'].sum()**

2.3 Διαχωρισμός Βάσει $ & Tonnes

Σύμφωνα με το project μας έπρεπε να διαχωρίσουμε τη συνολική τιμή του τζίρου στις δυο ακόλουθες μονάδες μέτρησης που υπήρχαν μέσα στο αρχείο μας σε ($ ή Tonnes). Έστω ότι θέλουμε να δημιουργήσουμε την δεύτερη γραφική μας παράσταση η οποία λέει:

Συνολική παρουσίαση του τζίρου (στήλη value) για κάθε χώρα (στις αντίστοιχες μονάδες μέτρησης).

Εφόσον έχουμε εκτελέσει τις εντολές grouped = df.groupby(['Country', 'Measure']) και totals = grouped['Value'].sum() που μας δίνουν τις στήλες που θέλουμε για τα δεδομένα μας δημιουργούμε δυο νέα DataFrame (*την ακόλουθη διαδικασία την υλοποιούμε και για τα άλλες γραφικές μας παραστάσεις) τα οποία είναι τα ακόλουθα **totals_usd = totals.reset_index()** και **totals_ton = totals.reset_index()** τα οποία είναι βασισμένα στο αρχικό DataFrame **totals.** Η μέθοδος **reset_index()** χρησιμοποιείται προκειμένου να επαναφέρουμε τον αριθμό ευρετηρίου (index) σε μια νέα στήλη και να δημιουργήσει ένα νέο αριθμό ευρετηρίου που αυξάνεται γραμμικά. Έπειτα για να ξεχωρίσουμε αν η μέτρηση του τζίρου είναι σε δολάρια ή τόνους εκτελούμε τις εντολές **totals_usd = totals_usd[totals_usd['Measure'] == '$']** και **totals_ton = totals_ton[totals_ton['Measure'] == 'Tonnes']** οι οποίες επιλέγουν μόνο τις γραμμές του DataFrame **totals_usd** και **totals_ton** όπου η τιμή της στήλης 'Measure' είναι ίση με $ ή Tonnes. Με αυτήν την ενέργεια δημιουργείται ένα φίλτρο όπου παραμένουν μόνο οι γραμμές που αντιστοιχούν στις μετρήσεις με $ και Tonnes.Τέλος με τις εντολές totals_usd = totals_usd[['Country', 'Value']] και totals_ton = totals_ton[['Country', 'Value']] δημιουργούμε δυο νέα DataFrames με όνομα **totals_usd** και **totals_ton** που περιέχουν μόνο τις στήλες Country και Value από τα DataFrames **totals_usd** και **totals_ton**. Με αυτόν τον τρόπο δημιουργείται ένα υποσύνολο των δεδομένων που περιλαμβάνει μόνο τις στήλες που αφορούν την χώρα (Country) και την τιμή του τζίρου ('Value') στις δυο μετρήσεις $ και Tonnes .

Ο τελικός κώδικας που παράγει τα δεδομένα που χρειάζονται για την δεύτερη γραφική παράσταση είναι ο εξής :

```
grouped = df.groupby(['Country', 'Measure'])

totals = grouped['Value'].sum()

totals_usd2 = totals.reset_index()

totals_usd2 = totals_usd2[totals_usd2['Measure'] == '$']

totals_usd2 = totals_usd2[['Country', 'Value']]
```

totals_ton2 = totals.reset_index()

totals_ton2 = totals_ton2[totals_ton2['Measure'] == 'Tonnes']

totals_ton2 = totals_ton2[['Country', 'Value']]

** Το μοτίβο του ακόλουθου κώδικα ακολουθείτε και για τα δεδομένα των άλλων γραφικών παραστάσεων αλλά με κάποιες διαφορετικές παραλλαγές που αναλύονται στην υπό-ενότητα 2.4 **

2.4 Παραλλαγές διαχείρισης δεδομένων

Παραλλαγή 1η

Λόγω του ότι στην 1η γραφική παράστασή χρειαζόταν ο συνολικός τζίρος ανά μηνά ενώ μέσα στο αρχείο μας η στήλη Date ήταν στην μορφή Ημέρα-Μήνας-Χρονιά εμείς έπρεπε να διαβάζουμε από την στήλη αυτή μόνο τον μήνα για αυτόν τον λόγω κάναμε τις ακόλουθες διεργασίες. Αρχικά δημιουργούμε μια νέα στήλη με όνομα 'Month' στο DataFrame μας **df['Month'].**Έπειτα για να καταφέρουμε να εισάγουμε στη στήλη αυτή τους μήνες από την στήλη Date εκτελούμε την εντολή df['Month'] = pd.to_datetime(df['Date'], format='%d/%m/%Y').dt.strftime('%B').Η μέθοδος **pd.to_datetime()** μετατρέπει τις τιμές της στήλης Date σε αντικείμενα ημερομηνίας, εφαρμόζοντας τη μορφή %d/%m/%Y. Στη συνέχεια, η μέθοδος **dt.strftime('%B')** μετατρέπει τα αντικείμενα ημερομηνίας σε συμβολοσειρές που αναπαριστούν τα ονόματα των μηνών που θέλουμε να χρησιμοποιήσουμε. Για την σωστή αναπαράσταση των μηνών δημιουργούμε μια λίστα με τα ονόματα των μηνών σε μια συγκεκριμένη σειρά δλδ **months_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December'].** Τέλος με την εντολή **df['Month'] = pd.Categorical(df['Month'], categories=months_order, ordered=True)** μετατρέπουμε τη στήλη 'Month' σε μια κατηγορική στήλη στο df . Η στήλη αυτή περιέχει κατηγορίες οι οποίες κατηγορίες είναι οι ονομασίες των μηνών που έχουν οριστεί στη λίστα **months_order**. Ορίζονται επίσης οι παράμετροι **categories=months_order** για να οριστούν οι κατηγορίες και **ordered=True** για να υποδηλωθεί ότι η κατηγορική στήλη έχει μια ορισμένη σειρά. Με αυτόν τον τρόπο, ο κώδικας δημιουργεί μια νέα στήλη Month στο **df**, η οποία περιέχει τις πληροφορίες για τον μήνα από τη στήλη Date.

Ο τελικός κώδικας που παράγει τα δεδομένα που χρειάζονται για την πρώτη γραφική παράσταση είναι ο εξής :

months_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']

df['Month']=pd.to_datetime(df['Date'],format='%d/%m/%Y').dt.strftime('%B')

df['Month']=pd.Categorical(df['Month'],categories=months_order,ordered=True)

totals = df.groupby(['Month', 'Measure'])['Value'].sum().reset_index()

totals_usd1 = totals[totals['Measure'] == '$'][['Month', 'Value']]

totals_ton1 = totals[totals['Measure'] == 'Tonnes'][['Month', 'Value']]

Παραλλαγή 2η

Επειδή στην 6η γραφική παράσταση μας ζητούσε τους 5 μήνες με τον μεγαλύτερο τζίρο πέρα από την μέθοδο groupby() και sum() που πρέπει να εκτελέσουμε στην στήλη Month πρέπει να ταξινομήσουμε το νέο dataframe προκειμένου να πάρουμε τους 5 μήνες με τον μεγαλύτερο τζίρο. Αυτό επιτυγχάνεται με την εντολή **totals_all = totals_all.sort_values(by='Value', ascending=False)** η οποία ταξινομεί το DataFrame totals_all με βάση τη στήλη 'Value' με φθίνουσα σειρά. Με αυτήν την ενέργεια, τα δεδομένα ταξινομούνται βάσει του συνολικού τζίρου για κάθε μήνα, με τον μεγαλύτερο τζίρο να βρίσκεται στην πρώτη θέση του DataFrame. Αφού ταξινομήσαμε τους μήνες βάσει τον τζίρου ήρθε η ωρα να επιλέξουμε του πέντε πρώτους μήνες , αυτό γίνεται ως εξής **top_5_months = totals_all.head(5).** Με αυτήν την εντολή επιλέγουμε τις πρώτες πέντε γραμμές του DataFrame **totals_all**, δηλαδή τους πέντε μήνες με τον υψηλότερο συνολικό τζίρο.

Ο τελικός κώδικας που παράγει τα δεδομένα που χρειάζονται για την έκτη γραφική παράσταση είναι ο εξής :

totals_all = df.groupby('Month')['Value'].sum().reset_index()

totals_all = totals_all.sort_values(by='Value', ascending=False)

top_5_months = totals_all.head(5)

Παραλλαγή 3η

Λόγω του ότι στη 7η γραφική παράσταση πρέπει να παρουσιάσουμε τις 5 κατηγορίες εμπορευμάτων με το μεγαλύτερο τζίρο για κάθε χωρά ξεχωριστά πρέπει να εκτελούσε ξανά την ίδια διαδικασία με την προηγούμενη παραλλαγή. Αρχικά δημιουργούμε ένα νέο dataframe με όνομα **totals_country_commodity** με την ακόλουθη εντολή **totals_country_commodity=df.groupby(['Country','Commodity'])['Value'].sum().reset_index().**Χρησιμοποιείται η μέθοδος groupby() για να ομαδοποιήσει τα δεδομένα βάσει των στηλών Country και Commodity. Στη συνέχεια, εφαρμόζεται η συνάρτηση άθροισης sum() στη στήλη Value για να υπολογιστεί το συνολικό άθροισμα της τιμής για κάθε χώρα και κατηγορία εμπορεύματος. Τέλος, με τη χρήση της μεθόδου **reset_index()** δημιουργείται ένας νέος αριθμός ευρετηρίου για το **df totals_country_commodity.**Για την ταξινόμηση κάνουμε τα ακόλουθα, αρχικά εκτελούμε την εντολή **totals_country_commodity=totals_country_commodity.sort_values(by=['Country', 'Value'], ascending=[True, False])** η οποία ταξινομεί το **df totals_country_commodity** με βάση τις στήλες **Country** και **Value**. Οι τιμές ταξινομούνται πρώτα με βάση την αλφαβητική σειρά της στήλης **Country** και έπειτα με βάση τον συνολικό τζίρο για κάθε χώρα, με φθίνουσα σειρά στη στήλη **Value.** Τέλος με την εκτέλεση της εντολής **top_commodity_country = totals_country_commodity.groupby('Country').head(5)**επιλέγουμε τις πρώτες πέντε γραμμές για κάθε χώρα από το **totals_country_commodity**. Αυτό σημαίνει ότι επιλέγονται οι πέντε κορυφαίες κατηγορίες εμπορευμάτων για κάθε χώρα, βάσει του υψηλότερου συνολικού τζίρου.

Ο τελικός κώδικας που παράγει τα δεδομένα που χρειάζονται για την εβδόμη γραφική παράσταση είναι ο εξής :

totals_country_commodity=df.groupby(['Country','Commodity'])['Value'].sum().reset_index()

totals_country_commodity=totals_country_commodity.sort_values(by=['Country', 'Value'], ascending=[True, False])

top_commodity_country=totals_country_commodity.groupby('Country').head(5)

Παραλλαγή 4η

Τέλος για τα δεδομένα τις τελευταίας γραφικής εκτελούμε τις ιδίες διαδικασίες με την παραλλαγή 3 μόνο που τώρα επειδή χρειαζόμαστε μόνο την μια μέρα με το μεγαλύτερο τζίρο για κάθε κατηγορία εμπορεύματος αντί να εκτελέσουμε την εντολή .head(5) που μας επιστρέφει τα πρώτα 5 στοιχεία από την στοιχισμένη στήλη θα εκτελέσουμε .head(1) που μας δίνει το πρώτο στοιχεί από την στοιχισμένη στήλη.

Ο τελικός κώδικας που παράγει τα δεδομένα που χρειάζονται για την ογδόη γραφική παράσταση είναι ο εξής :

totals_commodity_date=df.groupby(['Commodity','Date'])['Value'].sum().reset_index()

totals_commodity_date=totals_commodity_date.sort_values(by=['Commodity', 'Value'], ascending=[True, False])

top_commodity_category=totals_commodity_date.groupby('Commodity').head(1)

3ο Κεφάλαιο – Βάση Δεδομένων SQL

3.1 Σύνδεση με την MySQL

Εφόσον καταφέραμε να διαχειριστούμε κατάλληλα τα δεδομένα του αρχείου CSV προκειμένου να υλοποιήσουμε τις γράφηκες παραστάσεις που μας ζητήθηκαν , ήρθε η ώρα όπου πρέπει να αποθηκεύσουμε τα δεδομένα αυτά που θα χρησιμοποιήσουμε για την δημιουργία των γραφικών μας παραστάσεων σε κατάλληλα Tables στο περιβάλλον της MySQL. Για να συνδεθούμε στο περιβάλλον αυτό πρέπει να εκτελέσομε τις ακόλουθες εντολές:

```
db = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="karagiannis"
)
mycursor = db.cursor()
```

Αρχικά χρησιμοποιούμε την βιβλιοθήκη **mysql.connector** για να επικοινωνήσει με το πρόγραμμα της MySQL. Έπειτα θέτουμε τα καταλληλα προσωπικά μας στοιχεία στις μεταβλητές (**host, user, passwd**) οπού με αυτή την ανάθεση μπορούμε να συνδεθούμε στον δικό μας server.

- Στην μεταβλητή **host** θέτουμε το όνομα του διακομιστή της βάσης δεδομένων μας όπου σε αυτήν την περίπτωση είναι ο τοπικός υπολογιστής και γράφουμε "**localhost".**
- Στην μεταβλητή **user** θέτουμε το όνομα του χρήστη της βάσης δεδομένων μας που θα χρησιμοποιηθεί για την σύνδεση και σε αυτήν την περίπτωση είναι ο ριζικός χρήστης και γράφουμε "**root".**
- Στην μεταβλητή **passwd** θέτουμε τον κωδικό πρόσβασης για τον παραπάνω χρήστη που έχουμε ορίσει εμείς οπου σε αυτή την περίπτωση είναι ο "**karagiannis".**

Για να καταφέρουμε ωστόσο να εισάγουμε τα δεδομένα μας σε κατάλληλα Tables πρέπει να δημιουργήσουμε και μια κατάλληλη βάσει στην οποία θα περιέχονται τα Tables αυτά. Για να το επιτεύξουμε αυτό, αρχικά δημιουργούμε ένα αντικείμενο κέρσορα (**mycursor**) που θα χρησιμοποιηθεί για την εκτέλεση εντολών SQL στη βάση δεδομένων. Έτσι εκτελώντας την εντολή

```
mycursor.execute("CREATE DATABASE Covid_19_Data")
```

Καταφέρνουμε και δημιουργούμε μια νέα βάση δεδομένων με το όνομα **Covid_19_Data** όπου μέσα σε αυτήν θα δημιουργήσουμε τα κατάλληλα Tables που χρειαζόμαστε. Επίσης στην κλάση της σύνδεσης προσθέτουμε μια νέα μεταβλητή **database** όπου σε αυτήν αναθέτουμε το όνομα της βάσεις που θέλουμε και συνδεόμαστε κατευθείαν σε αυτήν όπου σε αυτήν την περίπτωση είναι η κλάση "**Covid_19_Data**". Έτσι γίνεται ως εξής η κλάση της σύνδεσης μας:

```
db=mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="karagiannis",
    database="Covid_19_Data"
)
```

3.2 Create & Insert

Προκειμένου να καταφέρουμε να εισάγουμε με ακρίβεια τα δεδομένα που θα απεικονίσουμε στις γραφικές παραστάσεις μας πρέπει να κάνουμε τα καταλληλά create insert για τα Tables της βάσεις που φτιάξαμε.

Create

Προκειμένου να καταφέρουμε να φτιάξουμε τα Tables μας από το περιβάλλον της Python εκτελούμε τις εξής εντολές για κάθε ερώτημα.

Πχ: Για το πρώτο ερώτημα που μας ζητάει την συνολική παρουσίαση του τζίρου (στήλη value) ανά μήνα (στις αντίστοιχες μονάδες μέτρησης), θα φτιάξουμε δυο ξεχωριστούς πίνακες που στον πρώτο θα απεικονίζονται οι μήνες με τον τζίρο σε δολάρια και στον δεύτερο οι μήνες με τον τζίρο σε τόνους. Ο κώδικας είναι ο ακόλουθος:

```
create_table_query1="CREATE TABLE Total_usd_per_month(Month
VARCHAR(255),DOLLARS FLOAT)"
mycursor.execute(create_table_query1)

create_table_query2 = "CREATE TABLE Total_ton_per_month (Month
VARCHAR(255),TONNES FLOAT)"
mycursor.execute(create_table_query2)
```

Την ίδια διαδικασία ακολουθούμε και για τα άλλα ερωτήματα ο κώδικας ακολουθεί παρακάτω:

Ερώτημα 2ο

```
create_table_query3 = "CREATE TABLE Total_usd_per_Country
(Country VARCHAR(255), DOLLARS FLOAT)"
mycursor.execute(create_table_query3)

create_table_query4 = "CREATE TABLE Total_ton_per_Country
(Country VARCHAR(255), TONNES FLOAT)"
mycursor.execute(create_table_query4)
```

Ερώτημα 3ο

```
create_table_query5 = "CREATE TABLE Total_usd_per_Transport
(Transport VARCHAR(255), DOLLARS FLOAT)"
mycursor.execute(create_table_query5)

create_table_query6 = "CREATE TABLE Total_ton_per_Transport
(Transport VARCHAR(255), TONNES FLOAT)"
mycursor.execute(create_table_query6)
```

Ερωτημα 4ο

```
create_table_query7 = "CREATE TABLE Total_usd_per_Weekday
(Weekday VARCHAR(255), DOLLARS FLOAT)"
mycursor.execute(create_table_query7)

create_table_query8 = "CREATE TABLE Total_ton_per_Weekday
(Weekday VARCHAR(255), TONNES FLOAT)"
mycursor.execute(create_table_query8)
```

Ερώτημα 5ο

```
create_table_query9 = "CREATE TABLE Total_usd_per_Commodity
(Commodity VARCHAR(255), DOLLARS FLOAT)"

mycursor.execute(create_table_query9)

create_table_query10 = "CREATE TABLE Total_ton_per_Commodity
(Commodity VARCHAR(255), TONNES FLOAT)"

mycursor.execute(create_table_query10)
```

Ερώτημα 6ο

create_table_query11 = "CREATE TABLE Top5_months (Month VARCHAR(255), Value FLOAT)"

mycursor.execute(create_table_query11)

Ερώτημα 7ο

create_table_query12 = "CREATE TABLE Top5_Commodities (Country VARCHAR(255), Commodity VARCHAR(255), Value FLOAT)"

mycursor.execute(create_table_query12)

Ερώτημα 8ο

create_table_query13 = "CREATE TABLE Top1_DATE (Commodity VARCHAR(255), Date VARCHAR(255), Value FLOAT)"

mycursor.execute(create_table_query13)

Στο τέλος κάθε script εκτελούμε την κάθε εντολή Create με την βοήθεια της εντολής **.execute** και του **mycursor** που ορίσαμε στην αρχή.

Insert

Εφόσον καταφέραμε και δημιουργήσαμε με επιτυχία του πίνακες ήρθε η στιγμή όπου πρέπει να εισάγομε στου πίνακες αυτούς τα καταλληλά δεδομένα. Αυτό το κάνουμε με τον κώδικα που ακολουθεί παρακάτω.

Πχ: Αφού καταφέραμε και δημιουργήσαμε δυο κατάλληλα Tables για τις δυο τιμές του τζίρου για κάθε μηνά, τώρα για την εισαγωγή των δεδομένων με την χρήση μιας δομής επανάληψης **for** επεξεργαζόμαστε κάθε γραμμή του DataFrame **totals_usd1** και **totals_ton1** που δημιουργήσαμε στο **2.3 Διαχωρισμός Βάσει $ & Tonnes.** Έπειτα για κάθε γραμμή, δημιουργούμε μια εντολή εισαγωγής (**INSERT**) SQL για να εισαγάγει τα δεδομένα στους πίνακες "**Total_usd_per_month**" και "**Total_ton_per_month**" της βάσης δεδομένων. Η μεταβλητή **insert_query1** και **insert_query2** περιέχουν το SQL ερώτημα εισαγωγής, ενώ η μέθοδος **mycursor.execute** εκτελεί τα ερωτήματα εισαγωγής για τις τρέχουσες γραμμές.

Ο κώδικας είναι ο ακόλουθος :

```
for _, row in totals_usd1.iterrows():
    insert_query1 = "INSERT INTO Total_usd_per_month (Month, DOLLARS) VALUES (%s, %s)"
    mycursor.execute(insert_query1, (row['Month'], float(row['Value'])))

for _, row in totals_ton1.iterrows():
    insert_query2 = "INSERT INTO Total_ton_per_month (Month, TONNES) VALUES (%s, %s)"
    mycursor.execute(insert_query2, (row['Month'], float(row['Value'])))
```

Την ίδια διαδικασία ακολουθούμε και για τα άλλα ερωτήματα ο κώδικας ακολουθεί παρακάτω:

Ερώτημα 2ο

```
for _, row in totals_usd2.iterrows():
    insert_query3 = "INSERT INTO Total_usd_per_Country (Country, DOLLARS) VALUES (%s, %s)"
    mycursor.execute(insert_query3, (row['Country'], float(row['Value'])))

for _, row in totals_ton2.iterrows():
    insert_query4 = "INSERT INTO Total_ton_per_Country (Country, TONNES) VALUES (%s, %s)"
    mycursor.execute(insert_query4, (row['Country'], float(row['Value'])))
```

Ερώτημα 3ο

```
for _, row in totals_usd3.iterrows():
    insert_query5 = "INSERT INTO Total_usd_per_Transport (Transport, DOLLARS) VALUES (%s, %s)"
    mycursor.execute(insert_query5, (row['Transport_Mode'], float(row['Value'])))

for _, row in totals_ton3.iterrows():
```

```
    insert_query6 = "INSERT INTO Total_ton_per_Transport (Transport, TONNES) VALUES (%s, %s)"
    mycursor.execute(insert_query6, (row['Transport_Mode'], float(row['Value'])))
```

Ερώτημα 4ο

```
for _, row in totals_usd4.iterrows():
    insert_query7 = "INSERT INTO Total_usd_per_Weekday (Weekday, DOLLARS) VALUES (%s, %s)"
    mycursor.execute(insert_query7, (row['Weekday'], float(row['Value'])))


for _, row in totals_ton4.iterrows():
    insert_query8 = "INSERT INTO Total_ton_per_Weekday (Weekday, TONNES) VALUES (%s, %s)"
    mycursor.execute(insert_query8, (row['Weekday'], float(row['Value'])))
```

Ερώτημα 5ο

```
for _, row in totals_usd5.iterrows():
    insert_query9 = "INSERT INTO Total_usd_per_Commodity (Commodity, DOLLARS) VALUES (%s, %s)"
    mycursor.execute(insert_query9, (row['Commodity'], float(row['Value'])))


for _, row in totals_ton5.iterrows():
    insert_query10 ="INSERT INTO Total_ton_per_Commodity (Commodity, TONNES) VALUES (%s, %s)"

mycursor.execute(insert_query10,(row['Commodity'],float(row['Value'])))
```

Ερώτημα 6ο

```
for _, row in top_5_months.iterrows():
    insert_query11 = "INSERT INTO Top5_months (Month, Value) VALUES (%s, %s)"
    mycursor.execute(insert_query11, (row['Month'], float(row['Value'])))
```

Ερώτημα 7ο

```
for country in top_commodity_country['Country'].unique():
    # Filter the data for the current country
    data = top_commodity_country[top_commodity_country['Country'] == country]

    # Iterate over the data for the current country and insert into the MySQL table
    for _, row in data.iterrows():
        insert_query12 = "INSERT INTO Top5_Commodities (Country, Commodity, Value) VALUES (%s, %s, %s)"
        mycursor.execute(insert_query12, (country, row['Commodity'], float(row['Value'])))
```

Ερώτημα 8ο

```
for commodity in top_commodity_category['Commodity'].unique():
    # Filter the data for the current country
    data = top_commodity_category[top_commodity_category['Commodity'] == commodity]

    # Iterate over the data for the current country and insert into the MySQL table
    for _, row in data.iterrows():
        insert_query13 = "INSERT INTO Top1_DATE (Commodity, Date, Value) VALUES (%s, %s, %s)"
        mycursor.execute(insert_query13, (commodity, row['Date'], float(row['Value'])))
```

3.3 Select

Τις εντολές αυτές δεν τις χρησιμοποιούμε τόσο για την βάση δεδομένων μας αλλά για να αποθηκεύσουμε τα αποτελέσματα των εντολών αυτών σε κατάλληλο αρχείο CSV.

Πχ: Προκειμένου να ανακτήσουμε τα δεδομένα μας από τους πίνακες **Total_usd_per_month** και **Total_ton_per_month** εκτελούμε την ακόλουθη εντολή.

```
select_query1 = "SELECT * FROM Total_usd_per_month"

mycursor.execute(select_query1)

data1 = mycursor.fetchall()


select_query2 = "SELECT * FROM Total_ton_per_month"

mycursor.execute(select_query2)

data2=mycursor.fetchall()
```

Με την μέθοδο **mycursor.execute** εκτελούμε το ερώτημα SQL και η μέθοδος **mycursor.fetchall** επιστρέφει όλα τα αποτελέσματα της ερώτησης ως μια λίστα από πλειάδες. Τα δεδομένα που ανακτήθηκαν αποθηκεύονται στη μεταβλητή **data1** και **data2** τα οποία δεδομένα αυτά θα χρησιμοποιηθούν στο μέλλον προκειμένου να τα αποθηκεύσουμε στο αρχείο csv που θα φτιάξουμε.

Την ίδια διαδικασία ακολουθούμε και για τα Tables της βάσεις μας και ο κώδικας μας είναι ο εξής:

Ερώτημα 2ο

```
select_query3 = "SELECT * FROM Total_usd_per_Country"

mycursor.execute(select_query3)

data3 = mycursor.fetchall()


select_query4 = "SELECT * FROM Total_ton_per_Country"

mycursor.execute(select_query4)

data4 = mycursor.fetchall()
```

Ερώτημα 3ο

```
select_query5 = "SELECT * FROM Total_usd_per_Transport"
mycursor.execute(select_query5)
data5 = mycursor.fetchall()

select_query6 = "SELECT * FROM Total_ton_per_Transport"
mycursor.execute(select_query6)
data6 = mycursor.fetchall()
```

Ερώτημα 4ο

```
select_query7 = "SELECT * FROM Total_usd_per_Weekday"
mycursor.execute(select_query7)
data7 = mycursor.fetchall()

select_query8 = "SELECT * FROM Total_ton_per_Weekday"
mycursor.execute(select_query8)
data8 = mycursor.fetchall()
```

Ερώτημα 5ο

```
select_query9 = "SELECT * FROM Total_usd_per_Commodity"
mycursor.execute(select_query9)
data9 = mycursor.fetchall()

select_query10 = "SELECT * FROM Total_ton_per_Commodity"
mycursor.execute(select_query10)
data10 = mycursor.fetchall()
```

Ερώτημα 6ο

```
select_query11 = "SELECT * FROM Top5_months"
mycursor.execute(select_query11)
data11 = mycursor.fetchall()
```

Ερώτημα 7ο

```
select_query12 = "SELECT * FROM Top5_Commodities"
mycursor.execute(select_query12)
data12 = mycursor.fetchall()
```

Ερώτημα 8ο

```
select_query13 = "SELECT * FROM Top1_DATE"
mycursor.execute(select_query13)
data13 = mycursor.fetchall()
```

4ο Κεφάλαιο – Αρχείο CSV

4.1 Δημιουργία & Αποθήκευση

Ένα από τα άλλα βασικά ερωτήματα που μας ανατέθηκαν στο project αυτό είναι να δημιουργήσουμε ένα αρχείο CSV στο οποίο θα αποθηκεύουμε τα δεδομένα των πινάκων που φτιάξαμε στην βάση δεδομένων μας. Αρχικά δημιουργούμε ένα νέο αρχείο CSV με όνομα '**Covid_19_Data.csv**' με την εντολή **csv_file = 'Covid_19_Data.csv'.** Έπειτα με τη χρήση της εντολής **open** ανοίγουμε το αρχείο μας για εγγραφή ('**w**'). Ο πίνακας δεδομένων γράφεται γραμμή προς γραμμή χρησιμοποιώντας τον **csv.writer .**Ο κώδικας είναι ο ακόλουθος:

```
with open(csv_file, 'w', newline='') as file:
    writer = csv.writer(file)
```

Για να καταφέρουμε να ξεχωρίσουμε κάθε πίνακα μεταξύ τους, θα εισάγουμε στο αρχείο μας μια γραμμή κεφαλίδων πριν από κάθε πίνακα με τα ονόματα των στηλών τους. **Πχ** για τον πίνακα Total_usd_per_month γράφουμε μια γραμμή κεφαλίδων με τα ονόματα των στηλών "Month" και "Dollars". Για να εισάγουμε τα δεδομένα τα οποία τα έχουμε αποθηκεύσει στην μεταβλητή data1 γράφουμε τις γραμμές δεδομένων για τον πίνακα "Total_usd_per_month" χρησιμοποιώντας την επανάληψη **for** και τη μέθοδο **writer.writerow.**

Έτσι ο κώδικας είναι :

```
with open(csv_file, 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow(['Month', 'Dollars'])
    for row1 in data1:
        writer.writerow(row1)
```

Μετά την εγγραφή των δεδομένων για τον πίνακα "Total_usd_per_month", προστίθεται μια κενή γραμμή για να χωρίσει τα δεδομένα των δύο πινάκων.

```
    writer.writerow([])
```

Την ίδια διαδικασία ακολουθούμε και για τους άλλους πίνακες που έχουμε δημιουργήσει , έτσι ο κώδικας που έχουμε είναι :

```
with open(csv_file, 'w', newline='') as file:
    writer = csv.writer(file)
```

Ερώτημα 1ο

```
    writer.writerow(['Month', 'Dollars'])
    for row1 in data1:
        writer.writerow(row1)
    writer.writerow([])
    writer.writerow(['Month', 'Tonnes'])
    for row2 in data2:
        writer.writerow(row2)
    writer.writerow([])
```

Ερώτημα 2ο

```
    writer.writerow(['Country', 'Dollars'])
    for row3 in data3:
        writer.writerow(row3)
    writer.writerow([])
    writer.writerow(['Country', 'Tonnes'])
    for row4 in data4:
        writer.writerow(row4)
    writer.writerow([])
```

Ερώτημα 3ο

```
    writer.writerow(['Transport', 'Dollars'])
    for row5 in data5:
```

```
        writer.writerow(row5)
    writer.writerow([])
    writer.writerow(['Transport', 'Tonnes'])
    for row6 in data6:
        writer.writerow(row6)
    writer.writerow([])
```

Ερώτημα 4ο

```
    writer.writerow(['Weekday', 'Dollars'])
    for row7 in data7:
        writer.writerow(row7)
    writer.writerow([])
    writer.writerow(['Weekday', 'Tonnes'])
    for row8 in data8:
        writer.writerow(row8)
    writer.writerow([])
```

Ερώτημα 5ο

```
    writer.writerow(['Commodity', 'Dollars'])
    for row9 in data9:
        writer.writerow(row9)
    writer.writerow([])
    writer.writerow(['Commodity', 'Tonnes'])
    for row10 in data10:
        writer.writerow(row10)
    writer.writerow([])
```

Ερώτημα 6ο

```
    writer.writerow(['Month', 'Dollars'])
    for row11 in data11:
        writer.writerow(row11)
```

```
    writer.writerow([])
```

Ερώτημα 7ο

```
    writer.writerow(['Country', 'Commodity', 'Value'])
    for row12 in data12:
        writer.writerow(row12)
    writer.writerow([])
```

Ερώτημα 8ο

```
    writer.writerow(['Commodity', 'Date', 'Value'])
    for row13 in data13:
        writer.writerow(row13)'''
```

5ο Κεφάλαιο – Γραφήματα & GUI

5.1 Δημιουργία γραφικών παραστάσεων

Εφόσον καταφέραμε και παράξαμε τα κατάλληλα δεδομένα που χρειαζόμασταν από το αρχικό CSV, ήρθε η ώρα που θα πρέπει να απεικονίσουμε αυτά τα δεδομένα σε γραφικές παραστάσεις .Αρχικά να για να αποφύγουμε την σύγχυση και τον συνωστισμό στον κώδικα μας ομαδοποιούμε τις γραφικές μας παραστάσεις σε ένα παράθυρο ώστε να έχουμε καλύτερη οπτική αυτών αλλά και να μπορούμε να συγκρίνουμε τα παραγόμενα δεδομένα μεταξύ τους. Αυτή η διαδικασία υλοποιήθηκε μέχρι κι το ερώτημα 6. Ο κώδικας για την υλοποίηση αυτή είναι ο ακόλουθος: **plt.subplots(1, 2, figsize=(12, 6)).** Η παράμετρος **(1, 2)** υποδεικνύει ότι θέλουμε ένα γράφημα με μία σειρά και δύο στήλες, ενώ η παράμετρος **figsize=(12, 6)** καθορίζει το μέγεθος του γραφήματος (12 ιντσών πλάτος και 6 ιντσών ύψος). Για να προσδιορίσουμε κατάλληλα τις τιμές που πρέπει να τοποθετήσουμε στα γραφήματα μας χρησιμοποιούμε τα DataFrames που δημιουργήσαμε στο κεφάλαιο 1.

Πχ: Έστω ότι θέλουμε να φτιάξουμε τις γραφικές παραστάσεις του ερωτήματος 1 τότε χρησιμοποιώντας τα df **totals_usd1** και **totals_ton1** για να προσδιοριστούν οι τιμές που θα εμφανιστούν στα διαγράμματα χρησιμοποιούμε επίσης τις εντολές **totals_usd1.plot()** και **totals_ton1.plot()** για να δημιουργήσουμε τα γραφήματα για τον τζίρο σε δολάρια και σε τόνους αντίστοιχα. Κάθε υποδιάγραμμα ρυθμίζεται με την χρήση αντίστοιχων εντολών **ax1.set_title()**, **ax2.set_xlabel()**, **ax1.set_ylabel()**, **ax2.set_title()**, **ax2.set_xlabel()** και **ax4.set_ylabel()** για την προσθήκη τίτλων και ετικετών στους άξονες των γραφημάτων. Τέλος, η εντολή **plt.show()** χρησιμοποιείται για να εμφανιστεί το γράφημα με τα υποδιαγράμματα.

Οι κώδικες για την δημιουργία των γραφημάτων είναι οι εξής:

Ερώτημα 1ο

```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))


totals_usd1.plot(kind='bar', x='Month', y='Value', ax=ax1, color='grey')
ax1.set_title('Total Turnover in USD by Month')
ax1.set_xlabel('Month')
ax1.set_ylabel('Total Turnover (USD)')
```

```
totals_ton1.plot(kind='bar', x='Month', y='Value', ax=ax2, color='maroon')
ax2.set_title('Total Turnover in Tonnes by Month')
ax2.set_xlabel('Month')
ax2.set_ylabel('Total Turnover (Tonnes)')

plt.show()
```

Ερώτημα 2ο

```
fig, (ax3, ax4) = plt.subplots(1, 2, figsize=(12, 6))

totals_usd2.plot(kind='bar', x='Country', y='Value', ax=ax3, color='grey')
ax3.set_title('Total Turnover in USD by Country')
ax3.set_xlabel('Country')
ax3.set_ylabel('Total Turnover (USD)')

totals_ton2.plot(kind='bar', x='Country', y='Value', ax=ax4, color='maroon')
ax4.set_title('Total Turnover in Tonnes by Country')
ax4.set_xlabel('Country')
ax4.set_ylabel('Total Turnover (Tonnes)')

plt.show()
```

Ερώτημα 3ο

```
fig, (ax5, ax6) = plt.subplots(1, 2, figsize=(12, 6))

totals_usd3.plot(kind='bar', x='Transport_Mode', y='Value', ax=ax5, color='grey')
ax5.set_title('Total Turnover in USD by Transport_Mode')
ax5.set_xlabel('Transport_Mode')
ax5.set_ylabel('Total Turnover (USD)')

totals_ton3.plot(kind='bar', x='Transport_Mode', y='Value', ax=ax6, color='maroon')
ax6.set_title('Total Turnover in Tonnes by Transport_Mode')
ax6.set_xlabel('Transport_Mode')
ax6.set_ylabel('Total Turnover (Tonnes)')

plt.show()
```

Ερώτημα 4ο

```
fig, (ax7, ax8) = plt.subplots(1, 2, figsize=(12, 6))

totals_usd4.plot(kind='bar', x='Weekday', y='Value', ax=ax7, color='grey')
ax7.set_title('Total Turnover in USD by Weekday')
ax7.set_xlabel('Weekday')
ax7.set_ylabel('Total Turnover (USD)')

totals_ton4.plot(kind='bar', x='Weekday', y='Value', ax=ax8, color='maroon')
ax8.set_title('Total Turnover in Tonnes by Weekday')
ax8.set_xlabel('Weekday')
```

```
ax8.set_ylabel('Total Turnover (Tonnes)')


plt.show()
```

Ερώτημα 5ο

```
fig, (ax9, ax10) = plt.subplots(1, 2, figsize=(12, 6))


totals_usd5.plot(kind='bar', x='Commodity', y='Value', ax=ax9, color='grey')
ax9.set_title('Total Turnover in USD by Commodity')
ax9.set_xlabel('Commodity')
ax9.set_ylabel('Total Turnover (USD)')


totals_ton5.plot(kind='bar', x='Commodity', y='Value', ax=ax10, color='maroon')
ax10.set_title('Total Turnover in Tonnes by Commodity')
ax10.set_xlabel('Commodity')
ax10.set_ylabel('Total Turnover (Tonnes)')


plt.show()
```

Ερώτημα 6ο

```
explode = (0.1, 0.0, 0.0, 0.0, 0.0)
fig, ax11 = plt.subplots(figsize=(8, 6))


top_5_months.plot(kind='pie', y='Value', labels=top_5_months['Month'], ax=ax11,colors=['blue', 'green', 'yellow', 'silver', 'red'], shadow=True, startangle=90, explode=explode, autopct='%1.1f%%')
ax11.set_title('Top 5 Months with the Highest Turnover')
ax11.set_ylabel('')
```

```
 plt.show()
```

Ερώτημα 7ο

```
    for country in top_commodity_country['Country'].unique():
        data = top_commodity_country[top_commodity_country['Country'] == country]


        fig, ax12 = plt.subplots(figsize=(8, 6))
        ax12.bar(data['Commodity'], data['Value'], width=0.2)
        ax12.set_title('Top 5 Commodities for {}'.format(country))
        ax12.set_xlabel('Commodity')
        ax12.set_ylabel('Total Turnover')


        plt.xticks(rotation=45)
        plt.show()
```

Ερώτημα 8ο

```
    for commodity in top_commodity_category['Commodity']:
        data = top_commodity_category[top_commodity_category['Commodity'] == commodity]
        fig, ax13 = plt.subplots(figsize=(8, 6))
        ax13.bar(data['Date'], data['Value'], width=0.2)
        ax13.set_title('Top Date for {}'.format(commodity))
        ax13.set_xlabel('Date')
        ax13.set_ylabel('Total Turnover')
        plt.xticks(rotation=0)
        plt.show()
```

5.2 Δημιουργία γραφικής διεπαφής (GUI)

Τέλος φτιάχνουμε ένα φιλικό περιβάλλον προς τον χρήστη έτσι ώστε να μπορεί να επεξεργάζεται με ευκολία και άνεση κάθε γραφική παράσταση αλλά και να μπορεί να συγκρίνει τα δεδομένα μεταξύ τους με έναν δυναμικό τρόπο. Για να καταφέρουμε να υλοποιήσουμε ένα τέτοιο περιβάλλον αρχικά δημιουργούμε το κύριο παράθυρο της διεπαφής μας με την χρήση της κλάσης **Tk()** από το Tkinter. Έπειτα ορίζουμε τις διαστάσεις του παραθύρου με την εντολή **root.geometry("1000x400")** και ένα κατάλληλο background με την εντολή **root.configure(background='#E0E0EE').** Επίσης ορίζουμε και έναν κατάλληλο τίτλο στο παράθυρο για την καλύτερη εμφάνιση στον χρήστη με την εντολή **root.title("Covid_19 Data").** Στο πρόγραμμά μας δημιουργούμε κατάλληλο label έτσι ώστε να μπορέσουμε να επεξηγήσουμε στον χρήστη την λειτουργία του προγράμματος που φτιάξαμε με την ακόλουθη εντολή : **Label(root, text=" ΘΕΤΟΥΜΕ ΤΟ ΚΕΙΜΕΝΟ ΜΑΣ ΕΔΩ ", font="Times 14", padx=25, pady=10, background='#E0E0EE').pack()**

Έπειτα δημιουργούμε δύο κουμπιά (buttons) με τις επιλογές "Continue" και "Exit".Το κουμπί "Continue" συνδέεται με μια συνάρτηση **open_new_window()** η οποία μας πάει σε ένα νέο παράθυρο οπού θα μπορούμε σε αυτό να επιλέξουμε εμείς την γραφική παράσταση που θέλουμε να δούμε. Το κουμπί "Exit" συνδέεται με μια συνάρτηση **close_window()** όπου όταν το πατάμε μας εμφανίζει κατάλληλο μήνυμα για το αν θέλουμε όντως να βγούμε από την εφαρμογή ενισχύοντας έτσι την δυναμική της εφαρμογής μας. Όταν επιλέγουμε το κουμπί continue μεταβαίνουμε σε νέο παράθυρο όπου πέρα από την παραμετροποίηση δημιουργούμε ένα dropdown bar όπου ο χρήστης επιλέγοντας κάποιες από τις ακόλουθε επιλογές: **(Chart 1, Chart 2, Chart 3, Chart 4, Chart 5, Chart 6, Chart 7, Chart 8)** οι οποίες συνδέονται με κατάλληλη συνάρτηση που περιέχουν τα κομμάτια του κώδικα που εμφανίζουν τις γραφικές παραστάσεις που θέλουμε, ο χρήστης θα μπορεί να επιλέξει και να εμφανίσει έτσι οποία γραφική παράστασή θέλει αυτός με όποια σειρά θέλει. Ο κώδικας για αυτό το κομμάτι είναι ο εξής:

```
chart_options = ["Chart 1", "Chart 2", "Chart 3", "Chart 4", "Chart 5", "Chart 6", "Chart 7", "Chart 8"]
    selected_chart = tk.StringVar()
    selected_chart.set(chart_options[0])  # Set the default selected option
chart_dropdown=OptionMenu(new_window,selected_chart,*chart_options)
    chart_dropdown.pack(padx=20, pady=20)
```

```
def handle_selection():
    selected_option = selected_chart.get()
    if selected_option == "Chart 1":
        show_chart1()
    elif selected_option == "Chart 2":
        show_chart2()
    elif selected_option == "Chart 3":
        show_chart3()
    elif selected_option == "Chart 4":
        show_chart4()
    elif selected_option == "Chart 5":
        show_chart5()
    elif selected_option == "Chart 6":
        show_chart6()
    elif selected_option == "Chart 7":
        show_chart7()
    elif selected_option == "Chart 8":
        show_chart8()
```

Ο κώδικας για την γραφική διεπαφή είναι ο εξής :

```
def show_chart1():
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

    totals_usd1.plot(kind='bar', x='Month', y='Value', ax=ax1, color='grey')
    ax1.set_title('Total Turnover in USD by Month')
    ax1.set_xlabel('Month')
    ax1.set_ylabel('Total Turnover (USD)')

    totals_ton1.plot(kind='bar', x='Month', y='Value', ax=ax2, color='maroon')
```

```
    ax2.set_title('Total Turnover in Tonnes by Month')
    ax2.set_xlabel('Month')
    ax2.set_ylabel('Total Turnover (Tonnes)')

    plt.show()


def show_chart2():
    fig, (ax3, ax4) = plt.subplots(1, 2, figsize=(12, 6))

    totals_usd2.plot(kind='bar', x='Country', y='Value', ax=ax3, color='grey')
    ax3.set_title('Total Turnover in USD by Country')
    ax3.set_xlabel('Country')
    ax3.set_ylabel('Total Turnover (USD)')

    totals_ton2.plot(kind='bar', x='Country', y='Value', ax=ax4, color='maroon')
    ax4.set_title('Total Turnover in Tonnes by Country')
    ax4.set_xlabel('Country')
    ax4.set_ylabel('Total Turnover (Tonnes)')
    plt.show()


def show_chart3():
    fig, (ax5, ax6) = plt.subplots(1, 2, figsize=(12, 6))
    totals_usd3.plot(kind='bar', x='Transport_Mode', y='Value', ax=ax5, color='grey')
    ax5.set_title('Total Turnover in USD by Transport_Mode')
    ax5.set_xlabel('Transport_Mode')
    ax5.set_ylabel('Total Turnover (USD)')

    totals_ton3.plot(kind='bar', x='Transport_Mode', y='Value', ax=ax6,
color='maroon')
    ax6.set_title('Total Turnover in Tonnes by Transport_Mode')
```

```
    ax6.set_xlabel('Transport_Mode')
    ax6.set_ylabel('Total Turnover (Tonnes)')

    plt.show()


def show_chart4():
    fig, (ax7, ax8) = plt.subplots(1, 2, figsize=(12, 6))
    totals_usd4.plot(kind='bar', x='Weekday', y='Value', ax=ax7, color='grey')
    ax7.set_title('Total Turnover in USD by Weekday')
    ax7.set_xlabel('Weekday')
    ax7.set_ylabel('Total Turnover (USD)')

    totals_ton4.plot(kind='bar', x='Weekday', y='Value', ax=ax8, color='maroon')
    ax8.set_title('Total Turnover in Tonnes by Weekday')
    ax8.set_xlabel('Weekday')
    ax8.set_ylabel('Total Turnover (Tonnes)')

    plt.show()


def show_chart5():
    fig, (ax9, ax10) = plt.subplots(1, 2, figsize=(12, 6))
    totals_usd5.plot(kind='bar', x='Commodity', y='Value', ax=ax9, color='grey')
    ax9.set_title('Total Turnover in USD by Commodity')
    ax9.set_xlabel('Commodity')
    ax9.set_ylabel('Total Turnover (USD)')

    totals_ton5.plot(kind='bar', x='Commodity', y='Value', ax=ax10, color='maroon')
    ax10.set_title('Total Turnover in Tonnes by Commodity')
    ax10.set_xlabel('Commodity')
    ax10.set_ylabel('Total Turnover (Tonnes)')
```

```
    plt.show()


def show_chart6():
    explode = (0.1, 0.0, 0.0, 0.0, 0.0)
    fig, ax11 = plt.subplots(figsize=(8, 6))

    top_5_months.plot(kind='pie', y='Value', labels=top_5_months['Month'], ax=ax11,
                      colors=['blue', 'green', 'yellow', 'silver', 'red'], shadow=True,
startangle=90, explode=explode,
                      autopct='%1.1f%%')
    ax11.set_title('Top 5 Months with the Highest Turnover')
    ax11.set_ylabel('')

    plt.show()


def show_chart7():
    for country in top_commodity_country['Country'].unique():
        data = top_commodity_country[top_commodity_country['Country'] == country]

        fig, ax12 = plt.subplots(figsize=(8, 6))
        ax12.bar(data['Commodity'], data['Value'], width=0.2)
        ax12.set_title('Top 5 Commodities for {}'.format(country))
        ax12.set_xlabel('Commodity')
        ax12.set_ylabel('Total Turnover')

        plt.xticks(rotation=45)

        plt.show()
```

```
def show_chart8():


    for commodity in top_commodity_category['Commodity']:

        data = top_commodity_category[top_commodity_category['Commodity'] == commodity]


        fig, ax13 = plt.subplots(figsize=(8, 6))

        ax13.bar(data['Date'], data['Value'], width=0.2)

        ax13.set_title('Top Date for {}'.format(commodity))

        ax13.set_xlabel('Date')

        ax13.set_ylabel('Total Turnover')

        plt.xticks(rotation=0)

        plt.show()



def open_new_window():

    def go_back():

        new_window.destroy()  # Close the current window

        root.deiconify()  # Restore visibility of the main window

    new_window = Toplevel(root)

    new_window.geometry("800x600")

    new_window.title("Charts")

    new_window.configure(background='#E0E0EE')

    Label(new_window, text="Καλώς ήρθατε στην ενότητα των γραφημάτων!", font="Times 22", padx=20, pady=10, background='#E0E0EE').pack()

    Label(new_window, text="Σε αυτήν την ενότητα μπορείτε να δείτε και να επεξεργαστείτε τα "

                           "γραφήματα που προέκυψαν έπειτα\n από σχολαστική μελέτη του αρχείου που μας ζητήθηκε."

                           "Παρακάτω σας παραθέτω τους τίτλους\n του κάθε γραφήματος για την περεταίρω διευκόλυνση σας."

          , font="Times 14", padx=30, pady=10, background='#E0E0EE').pack()

    Label(new_window, text="Chart1:Συνολική παρουσίαση του τζίρου ανά μήνα.\n"
```

```
                "Chart2:Συνολική παρουσίαση του τζίρου για κάθε χώρα.\n"
                "Chart3:Συνολική παρουσίαση του τζίρου για κάθε μέσο μεταφοράς.\n"
                "Chart4:Συνολική παρουσίαση του τζίρου για κάθε μέρα της εβδομάδας.\n"
                "Chart5:Συνολική παρουσίαση του τζίρου για κάθε κατηγορία εμπορεύματος.\n"
                "Chart6:Παρουσίαση των 5 μηνών με το μεγαλύτερο τζίρο,ανεξαρτήτως\n μέσου μεταφοράς και είδους ανακυκλώσιμων ειδών.\n"
                "Chart7:Παρουσίαση των 5 κατηγοριών εμπορευμάτων με το μεγαλύτερο τζίρο,για κάθε χώρα.\n"
                "Chart8:Παρουσίαση της ημέρας με το μεγαλύτερο τζίρο, για κάθε κατηγορία εμπορεύματος.\n"
        , font="Times 14", padx=25, pady=10, background='#E0E0EE').pack()


    chart_options = ["Chart 1", "Chart 2", "Chart 3", "Chart 4", "Chart 5", "Chart 6", "Chart 7", "Chart 8"]
    selected_chart = tk.StringVar()
    selected_chart.set(chart_options[0])  # Set the default selected option


    chart_dropdown=OptionMenu(new_window,selected_chart,*chart_options)
    chart_dropdown.pack(padx=20, pady=20)


    def handle_selection():
        selected_option = selected_chart.get()
        if selected_option == "Chart 1":
            show_chart1()
        elif selected_option == "Chart 2":
            show_chart2()
        elif selected_option == "Chart 3":
            show_chart3()
        elif selected_option == "Chart 4":
            show_chart4()
        elif selected_option == "Chart 5":
```

```
            show_chart5()
        elif selected_option == "Chart 6":
            show_chart6()
        elif selected_option == "Chart 7":
            show_chart7()
        elif selected_option == "Chart 8":
            show_chart8()

chart_button=tk.Button(new_window,text="ShowChart",command=handle_selection, background='#EEEEE9')
    chart_button.pack(padx=20, pady=20)
    back_button = tk.Button(new_window, text="Back", command=go_back, background='#EEEEE9')
    back_button.pack(padx=20, pady=20)



def close_window():
    if messagebox.askokcancel("Close", "Are you sure you want to close the window?"):
        root.destroy()  # Close the current window



root = tk.Tk()
root.geometry("1000x400")
root.title("Covid_19 Data")
root.configure(background='#E0E0EE')



Label(root, text="Covid-19 X Python ?", font="Times 22", padx=20, pady=10, background='#E0E0EE').pack()
Label(root, text="Στο project αυτό, αναλύσαμε ένα αρχείο CSV για την επίδραση του COVID-19 στο εμπόριο.\n"
                 "Χρησιμοποιώντας κατάλληλα ορίσαματα της γλώσσας Python,καταφέραμε να διαβάσαμε τα δεδομένα\n"
```

" και δημιουργήσαμε τα απαραιτητα γραφήματα για την οπτική αναπαράσταση τους.\n"

"Επειτα χρησιμοποιώντασ το περιβάλλων τησ Mysql, καταφέραμε να αποθηκεύσουμε τα δεδομένα \n"

" σε πίνακες για την καλύτερη διαχείρηση τους για περεταίρω εργασίες στο αμμεσο μέλλον.\n"

"Τα γραφήματα οπωσ θα δείται και εσείς μας αποκάλυψαν μια εικόνα της επίδρασης του COVID-19 στο παγκόσμιο εμπόριο.\n"

"Εάν θέλετε να δείτε τα γραφήματα που πρόεκυψαν παρακαλώ πατείστε το κουμπι Continue, αλλιώς μπορείτε να βγείτε \n"

"από την εφαρμογή πατώντας Exit. Σας ευχαριστώ πολύ!"

, font="Times 14", padx=25, pady=10, background='#E0E0EE').pack()

open_button=tk.Button(root,text="Continue",command=open_new_window,background='#E0E0EE')

open_button.pack(padx=50, pady=20)

close_button=tk.Button(root,text="Exit",command=close_window,background='#E0E0EE')

close_button.pack(padx=0, pady=20)

root.mainloop()

6ο Κεφάλαιο – Αποτελέσματα Project

6.1 Εκτέλεση SQL ερωτημάτων.

Εφόσον δημιουργούμε την βάση δεδομένων μας και δημιουργήσουμε τα Tables στο περιβάλλον της MySQL έχουμε τα εξής αποτελέσματα:

```
mysql> use covid_19_data;
Database changed
mysql> show tables;
+--------------------------+
| Tables_in_covid_19_data  |
+--------------------------+
| top1_date                |
| top5_commodities         |
| top5_months              |
| total_ton_per_commodity  |
| total_ton_per_country    |
| total_ton_per_month      |
| total_ton_per_transport  |
| total_ton_per_weekday    |
| total_usd_per_commodity  |
| total_usd_per_country    |
| total_usd_per_month      |
| total_usd_per_transport  |
| total_usd_per_weekday    |
+--------------------------+
13 rows in set (0.02 sec)
```

Για την εκτέλεση των εντολών Select έχουμε:

Ερώτημα 1ο

```
mysql> select * from total_usd_per_month;
+-----------+--------------+
| Month     | DOLLARS      |
+-----------+--------------+
| January   | 244882000000 |
| February  | 244682000000 |
| March     | 278497000000 |
| April     | 266910000000 |
| May       | 284768000000 |
| June      | 273339000000 |
| July      | 280879000000 |
| August    | 253427000000 |
| September | 259601000000 |
| October   | 285419000000 |
| November  | 293291000000 |
| December  | 272102000000 |
+-----------+--------------+
12 rows in set (0.00 sec)

mysql> select * from total_ton_per_month;
+-----------+----------+
| Month     | TONNES   |
+-----------+----------+
| January   | 20022000 |
| February  | 25850000 |
| March     | 28334000 |
| April     | 24862000 |
| May       | 26527000 |
| June      | 27495000 |
| July      | 28326000 |
| August    | 25285000 |
| September | 24033000 |
| October   | 28009000 |
| November  | 28244000 |
| December  | 25410000 |
+-----------+----------+
12 rows in set (0.00 sec)
```

Ερώτημα 2ο

```
mysql> select * from total_usd_per_country;
+-----------------------------+---------------+
| Country                     | DOLLARS       |
+-----------------------------+---------------+
| All                         | 2315200000000 |
| Australia                   |  107686000000 |
| China                       |  282650000000 |
| East Asia (excluding China) |  116556000000 |
| European Union (27)         |   26644000000 |
| Japan                       |   23155000000 |
| Total (excluding China)     |  291991000000 |
| United Kingdom              |   21591000000 |
| United States               |   52320000000 |
+-----------------------------+---------------+
9 rows in set (0.00 sec)

mysql> select * from total_ton_per_country;
+-----------------------------+-----------+
| Country                     | TONNES    |
+-----------------------------+-----------+
| All                         | 185349000 |
| China                       | 119573000 |
| East Asia (excluding China) |   6137000 |
| United States               |   1338000 |
+-----------------------------+-----------+
4 rows in set (0.00 sec)
```

Ερώτημα 3ο

```
mysql> select * from total_usd_per_transport;
+-----------+---------------+
| Transport | DOLLARS       |
+-----------+---------------+
| Air       |  132602000000 |
| All       | 2436790000000 |
| Sea       |  668400000000 |
+-----------+---------------+
3 rows in set (0.00 sec)

mysql> select * from total_ton_per_transport;
+-----------+-----------+
| Transport | TONNES    |
+-----------+-----------+
| All       | 312397000 |
+-----------+-----------+
1 row in set (0.00 sec)
```

Ερώτημα4ο

```
mysql> select * from total_usd_per_weekday;
+-----------+--------------+
| Weekday   | DOLLARS      |
+-----------+--------------+
| Sunday    | 309677000000 |
| Monday    | 566712000000 |
| Tuesday   | 520469000000 |
| Wednesday | 517338000000 |
| Thursday  | 532286000000 |
| Friday    | 533090000000 |
| Saturday  | 258225000000 |
+-----------+--------------+
7 rows in set (0.00 sec)

mysql> select * from total_ton_per_weekday;
+-----------+----------+
| Weekday   | TONNES   |
+-----------+----------+
| Sunday    | 43239000 |
| Monday    | 48527000 |
| Tuesday   | 42585000 |
| Wednesday | 44998000 |
| Thursday  | 43047000 |
| Friday    | 47760000 |
| Saturday  | 42241000 |
+-----------+----------+
7 rows in set (0.00 sec)
```

Ερώτημα 5ο

```
mysql> select * from total_usd_per_commodity;
+--------------------------------+---------------+
| Commodity                      | DOLLARS       |
+--------------------------------+---------------+
| All                            | 2386670000000 |
| Electrical machinery and equip |   51554000000 |
| Fish, crustaceans, and molluscs |   15445000000 |
| Fruit                          |   22197000000 |
| Logs, wood, and wood articles  |   50381000000 |
| Meat and edible offal          |   78512000000 |
| Mechanical machinery and equip |   72603000000 |
| Milk powder, butter, and cheese |  157284000000 |
| Non-food manufactured goods    |  403154000000 |
+--------------------------------+---------------+
9 rows in set (0.00 sec)

mysql> select * from total_ton_per_commodity;
+--------------------------------+-----------+
| Commodity                      | TONNES    |
+--------------------------------+-----------+
| Fish, crustaceans, and molluscs |   1832000 |
| Logs, wood, and wood articles  | 264402000 |
| Meat and edible offal          |  10372000 |
| Milk powder, butter, and cheese |  35791000 |
+--------------------------------+-----------+
4 rows in set (0.00 sec)
```

Ερώτημα 6ο

```
mysql> select * from top5_months;
+----------+--------------+
| Month    | Value        |
+----------+--------------+
| November | 293319000000 |
| October  | 285447000000 |
| May      | 284795000000 |
| July     | 280907000000 |
| March    | 278525000000 |
+----------+--------------+
5 rows in set (0.00 sec)
```

Ερώτημα 7ο

```
mysql> select * from top5_commodities;
+---------------------------+--------------------------------+--------------+
| Country                   | Commodity                      | Value        |
+---------------------------+--------------------------------+--------------+
| All                       | All                            | 1603470000000 |
| All                       | Non-food manufactured goods    |  403154000000 |
| All                       | Milk powder, butter, and cheese |   98779100000 |
| All                       | Mechanical machinery and equip |   57567000000 |
| All                       | Meat and edible offal          |   51212800000 |
| Australia                 | All                            |  107686000000 |
| China                     | All                            |  182406000000 |
| China                     | Milk powder, butter, and cheese |   31223500000 |
| China                     | Logs, wood, and wood articles  |   18102800000 |
| China                     | Electrical machinery and equip |   16478000000 |
| China                     | Meat and edible offal          |   15465300000 |
| East Asia (excluding China) | All                          |   89245000000 |
| East Asia (excluding China) | Milk powder, butter, and cheese |   27317100000 |
| European Union (27)       | All                            |   26644000000 |
| Japan                     | All                            |   23155000000 |
| Total (excluding China)   | All                            |  291991000000 |
| United Kingdom            | All                            |   21591000000 |
| United States             | All                            |   40477000000 |
| United States             | Meat and edible offal          |   11844300000 |
+---------------------------+--------------------------------+--------------+
19 rows in set (0.00 sec)
```

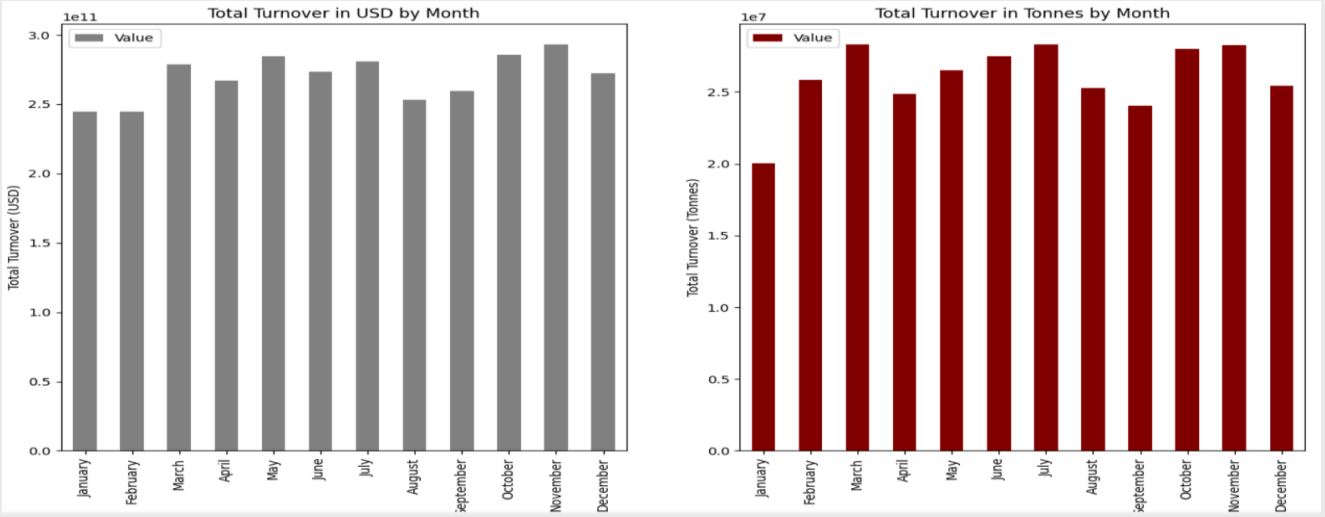
Ερώτημα 8ο

```
mysql> select * from top1_date;
+--------------------------------+------------+------------+
| Commodity                      | Date       | Value      |
+--------------------------------+------------+------------+
| All                            | 26/11/2021 | 2227000000 |
| Electrical machinery and equip | 06/11/2020 |   96000000 |
| Fish, crustaceans, and molluscs | 25/09/2017 |   21003000 |
| Fruit                          | 03/05/2020 |   63000000 |
| Logs, wood, and wood articles  | 30/07/2021 |   97412000 |
| Meat and edible offal          | 21/05/2017 |   99014000 |
| Mechanical machinery and equip | 02/10/2018 |  143000000 |
| Milk powder, butter, and cheese | 04/12/2019 |  419087000 |
| Non-food manufactured goods    | 26/06/2020 |  565000000 |
+--------------------------------+------------+------------+
9 rows in set (0.00 sec)
```

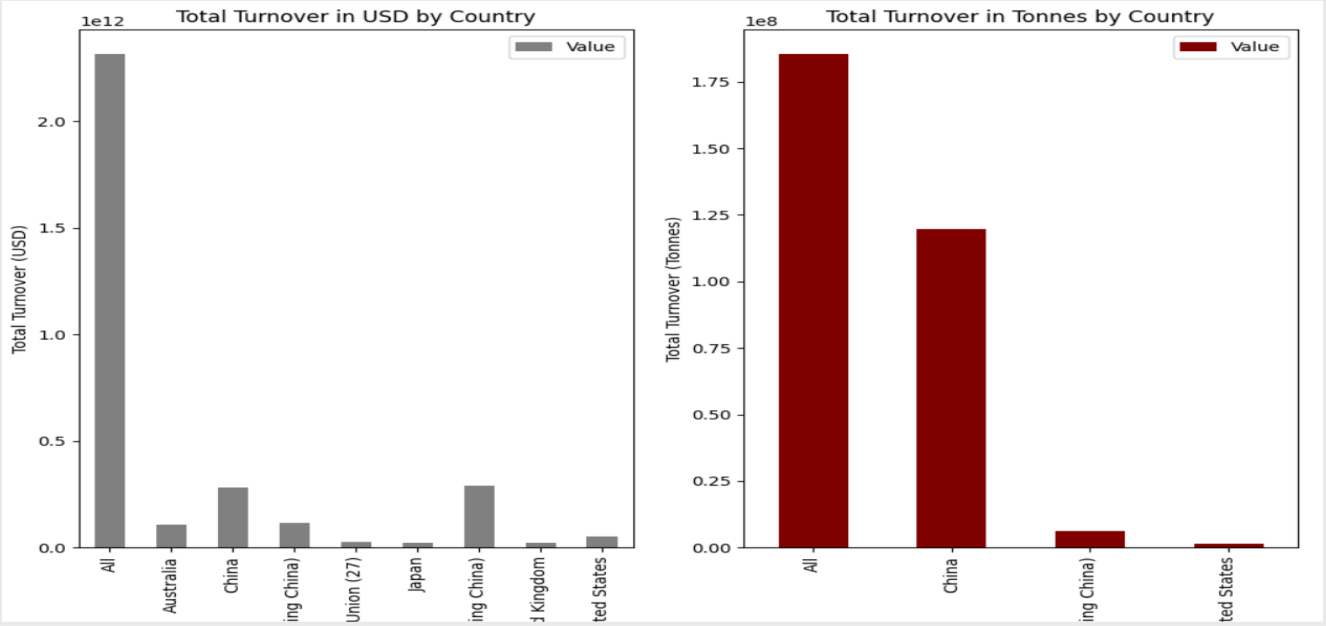
6.2 Γραφικές παραστάσεις

Οι γραφικές μας παραστάσεις είναι οι εξής :

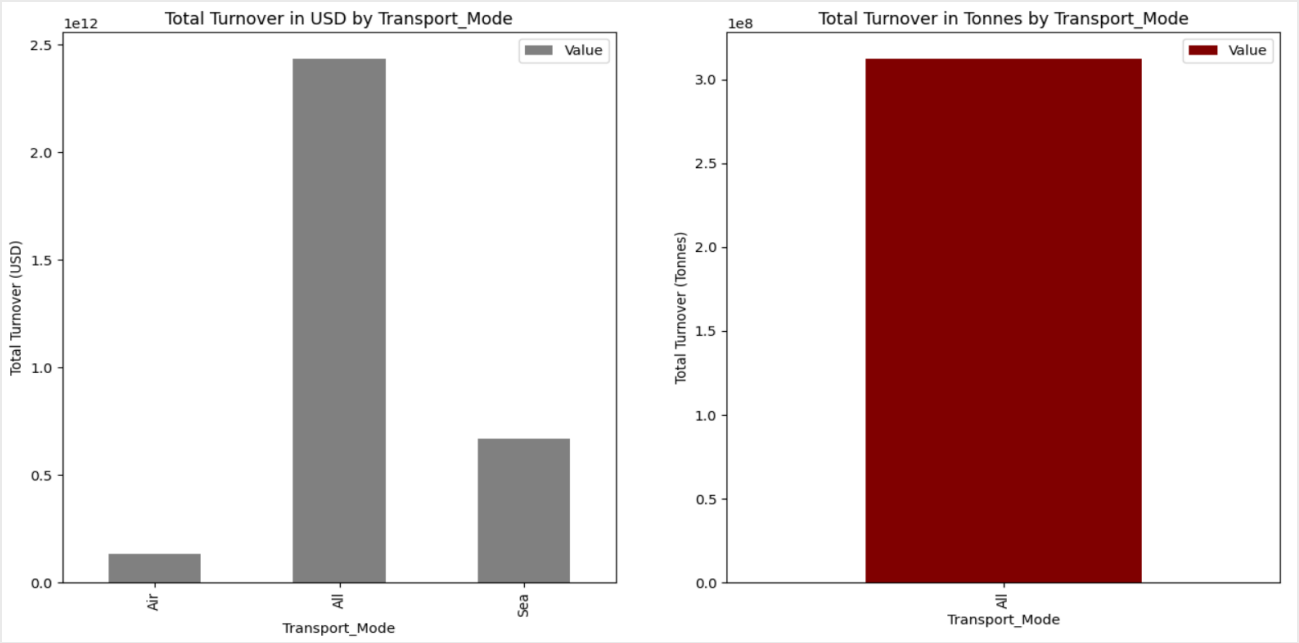
Ερώτημα 1ο



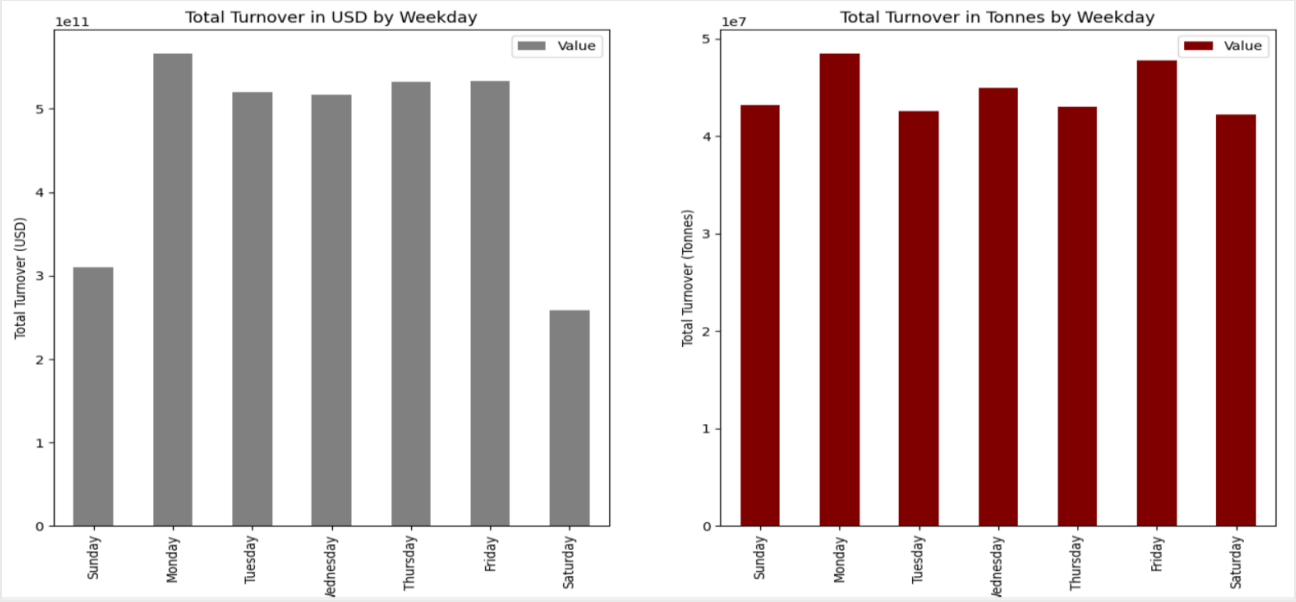
Ερώτημα 2ο



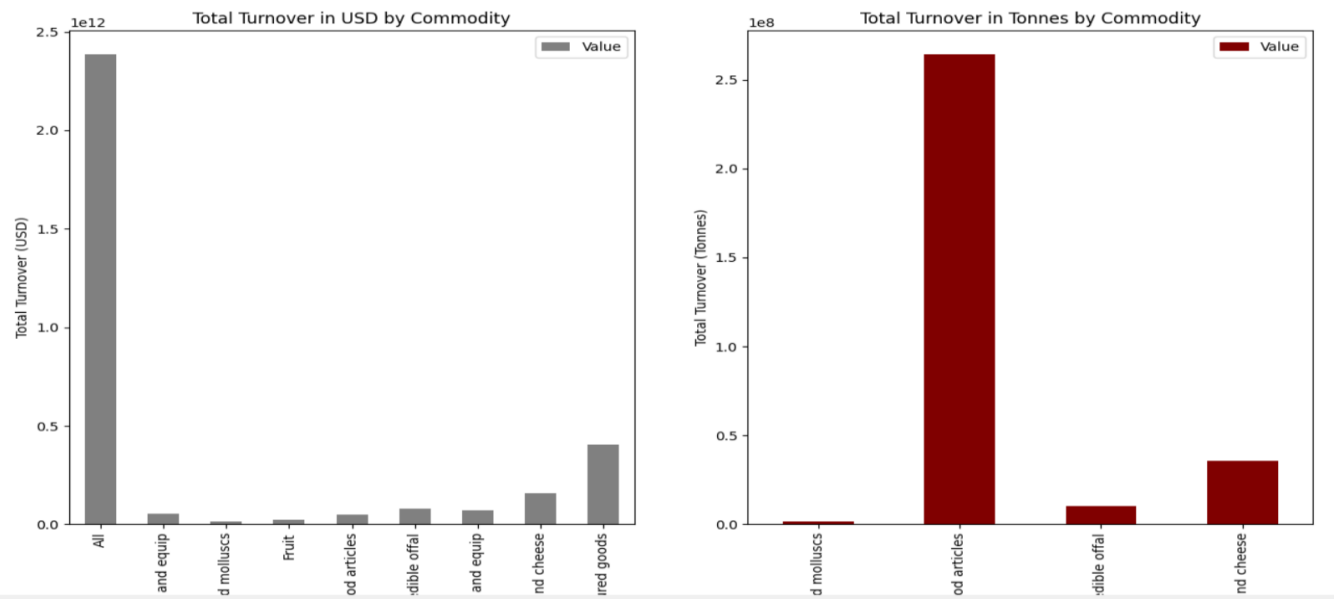
Ερώτημα 3ο



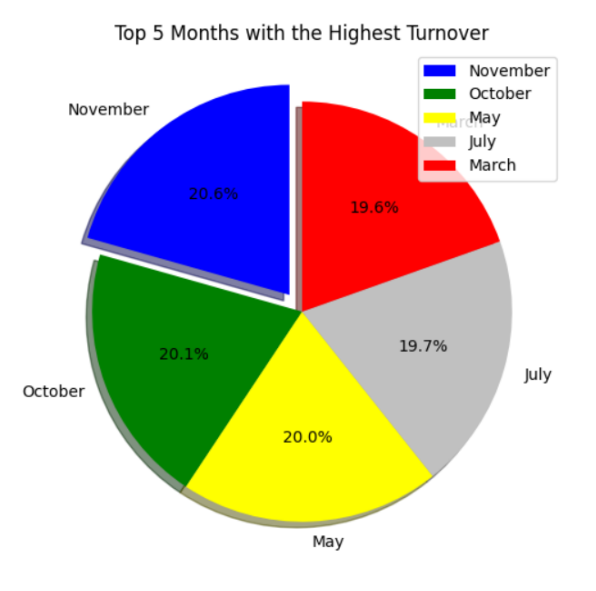
Ερώτημα 4ο



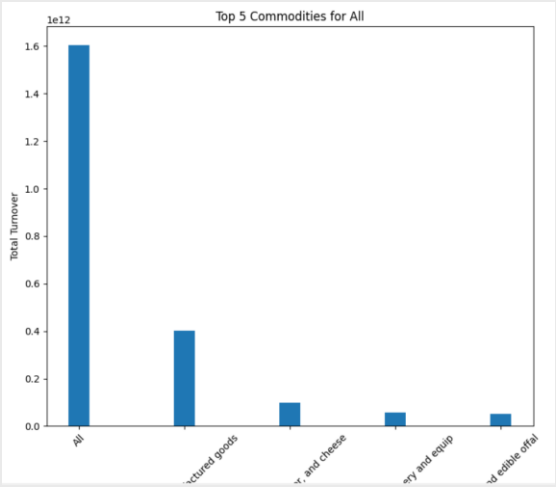
Ερώτημα 5ο



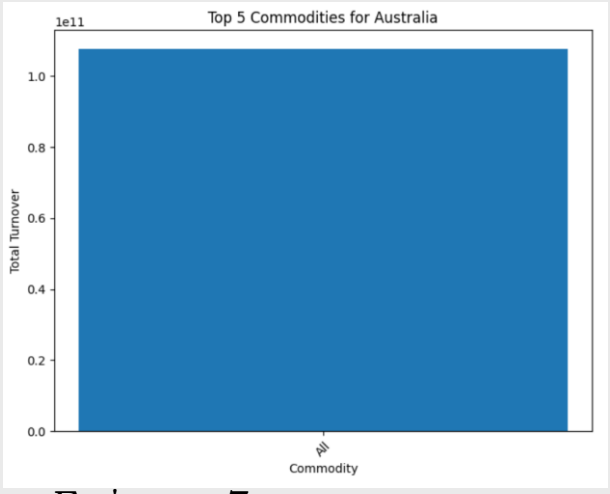
Ερώτημα 6ο



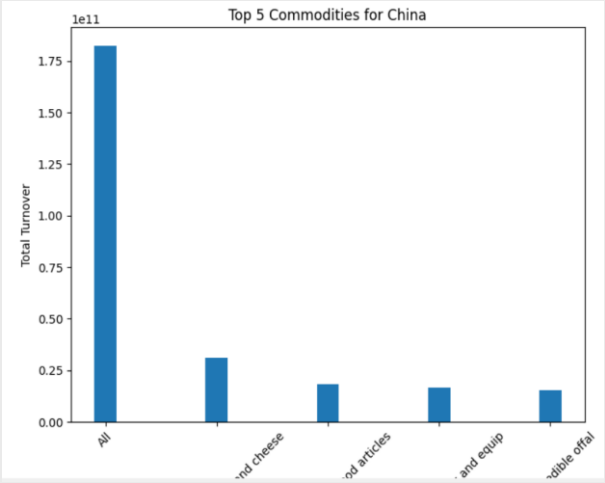
Ερώτημα 7ο



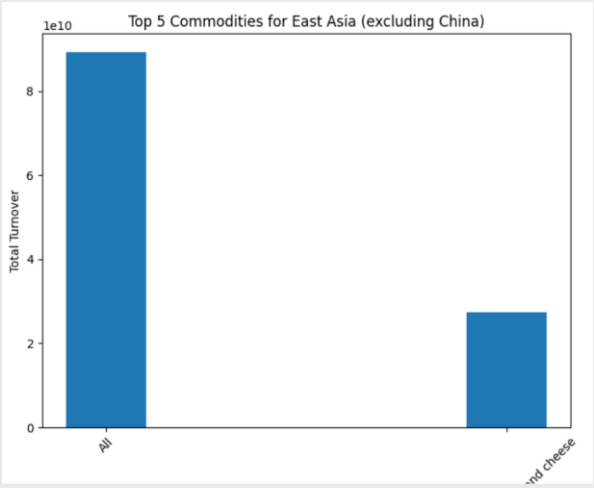
Ερώτημα 7ο



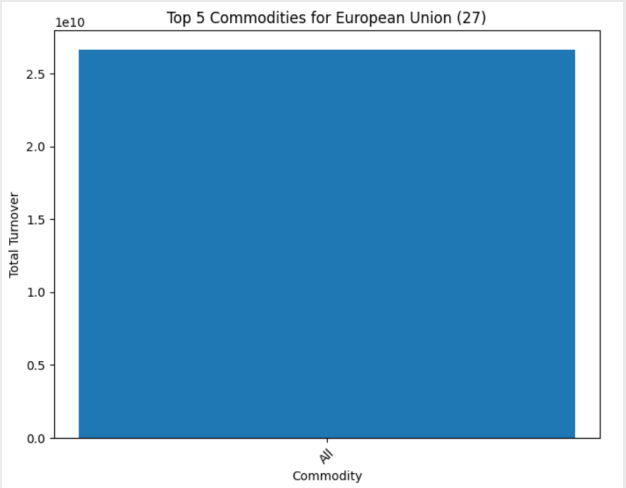
Ερώτημα 7ο



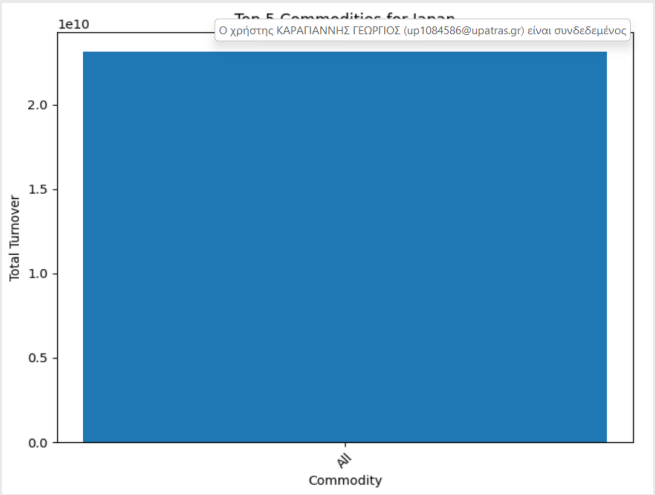
Ερώτημα 7ο



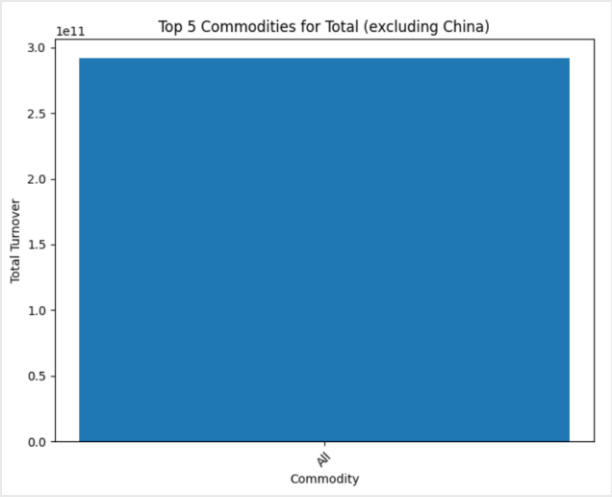
Ερώτημα 7ο



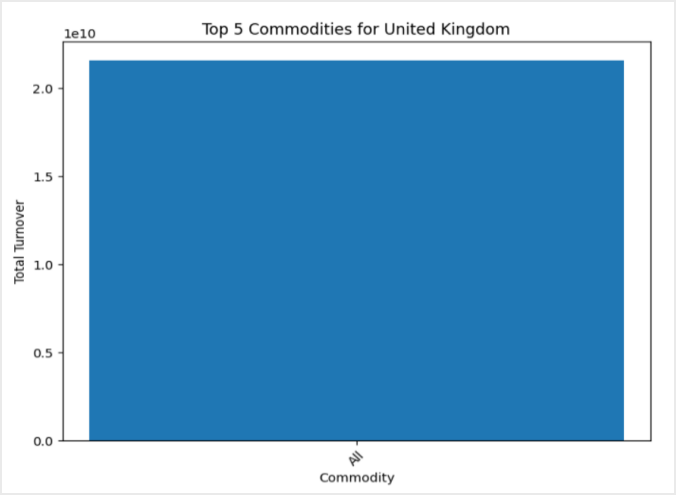
Ερώτημα 7ο



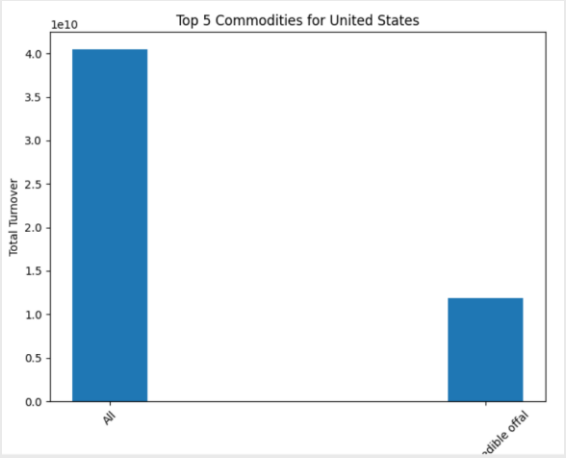
Ερώτημα 7ο



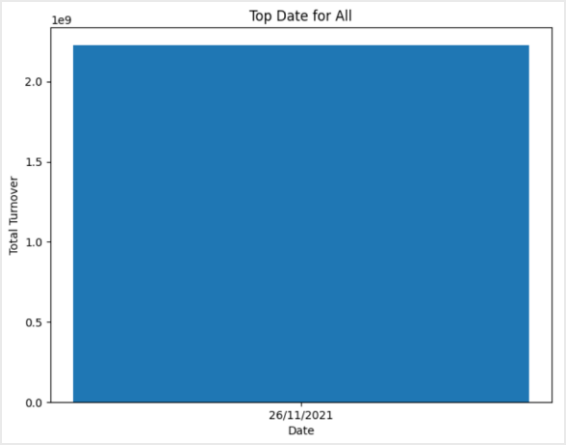
Ερώτημα 7ο



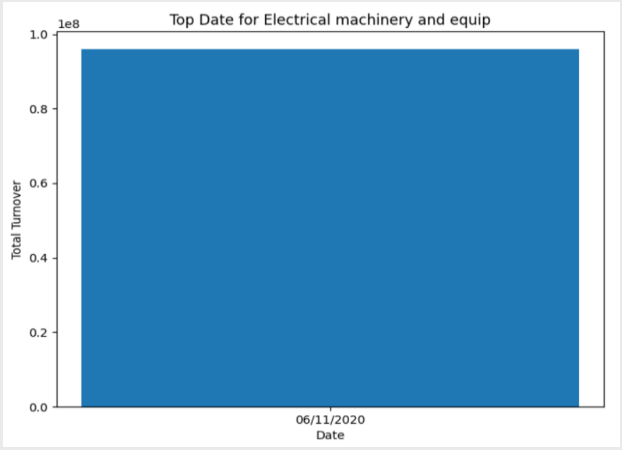
Ερώτημα 7ο



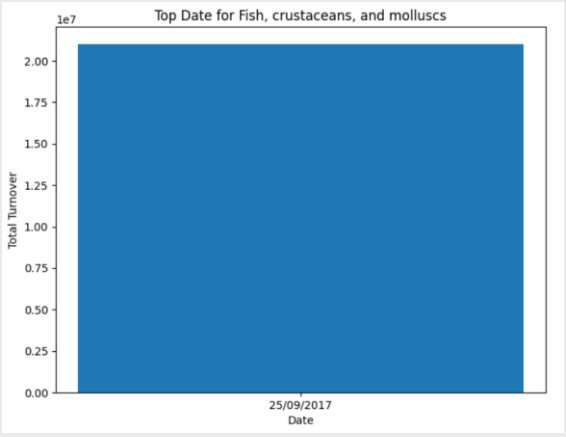
Ερώτημα8ο



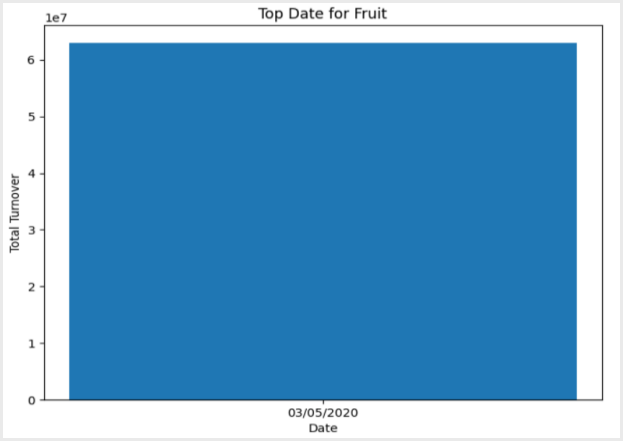
Ερώτημα8ο



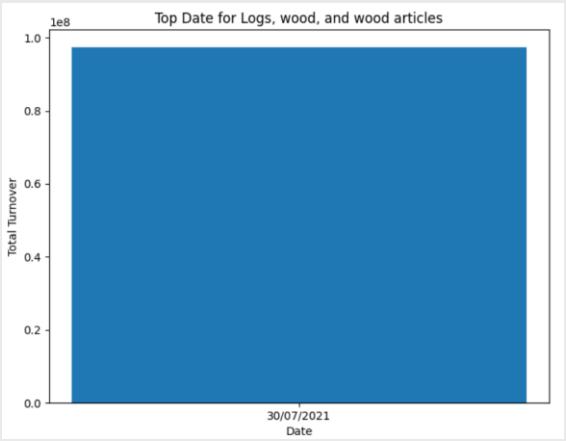
Ερώτημα 8ο



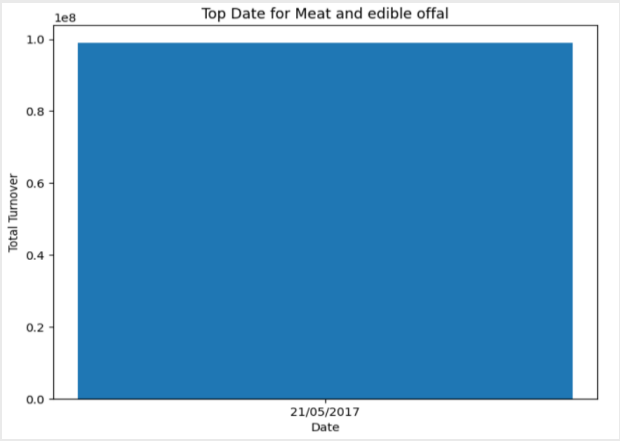
Ερώτημα 8ο



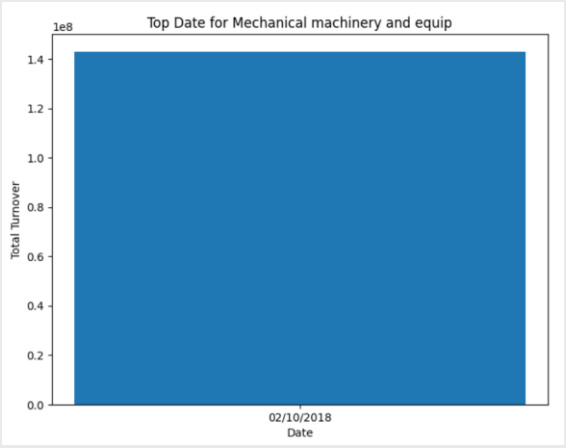
Ερώτημα 8ο



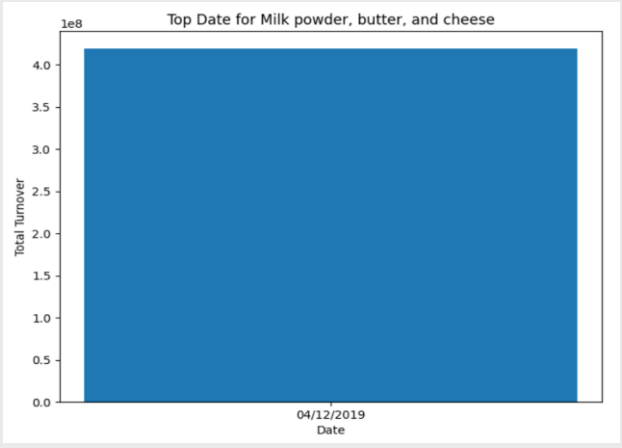
Ερώτημα 8ο



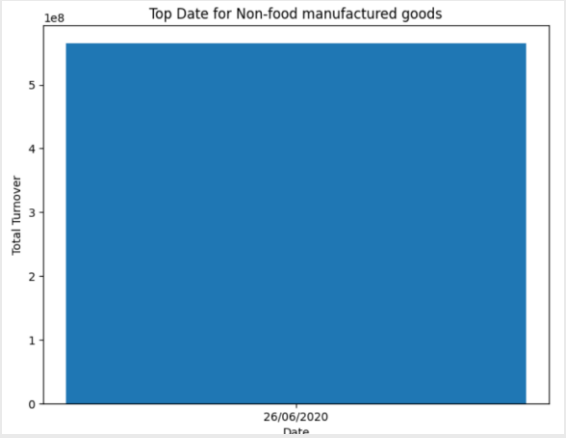
Ερώτημα 8ο



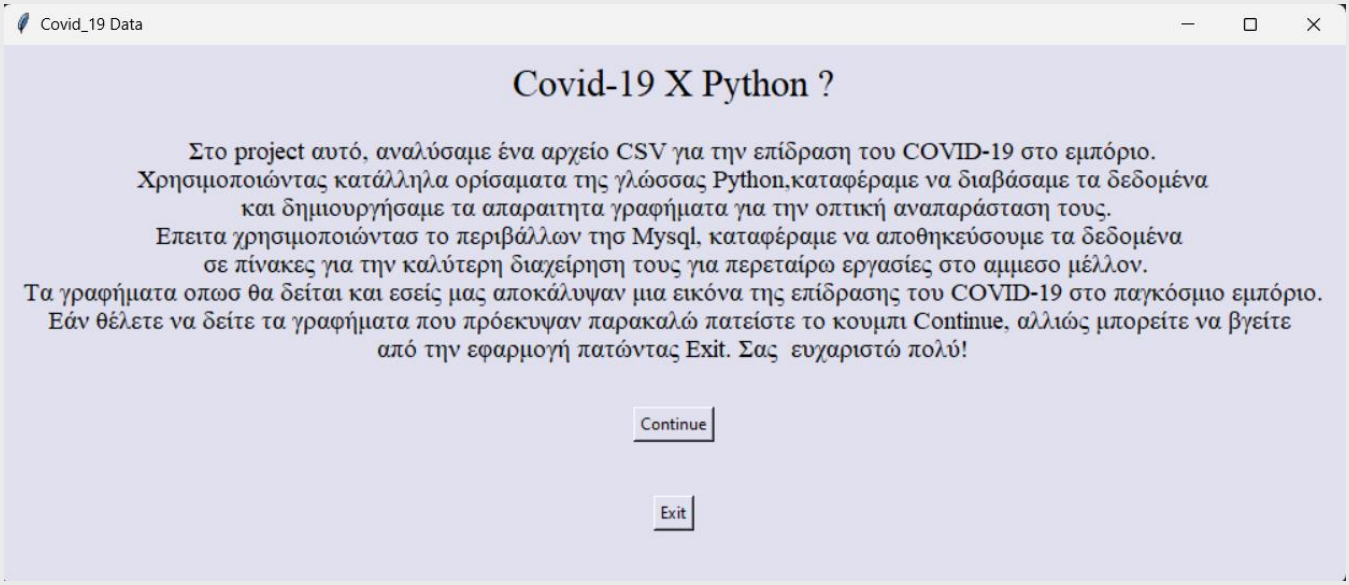
Ερώτημα 8ο

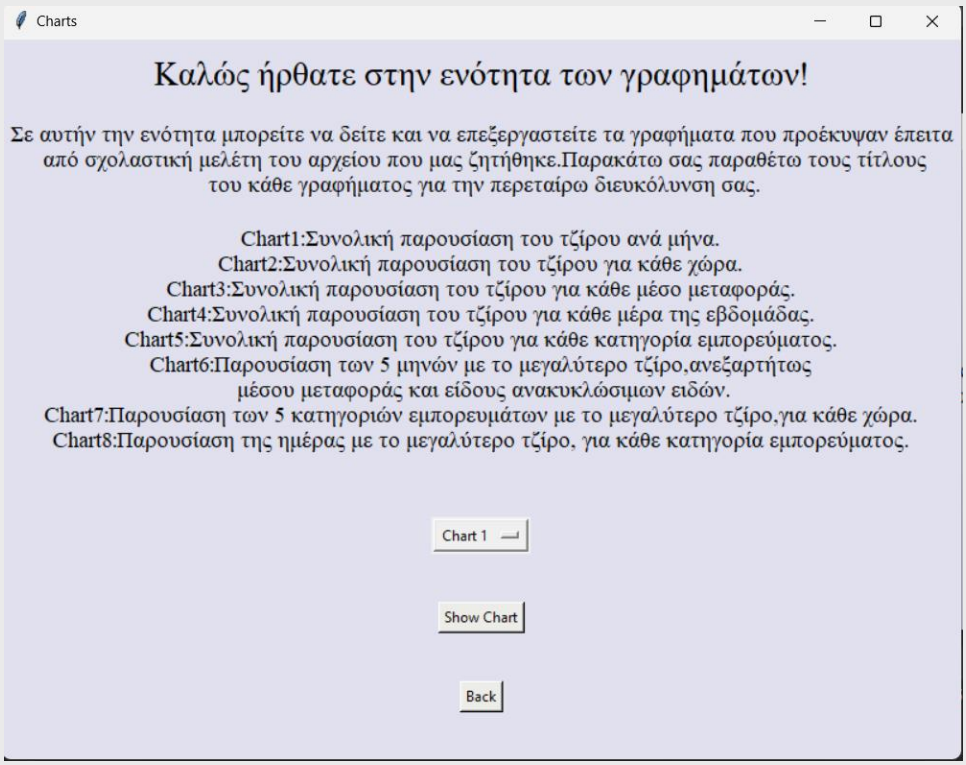


Ερώτημα 8ο



6.3 Περιβάλλον GUI





7ο Κεφάλαιο –Παραδοχές

Στα ερωτήματα 6, 7 και 8 λόγο λάθους κατανόησης των εκφωνήσεων οι γραφικές παραστάσεις μας δεν εμπεριέχουν τα δεδομένα που ζητούνται με βάσει τις δυο διαφορετικές μονάδες μέτρησης που έχουμε, δηλαδή δολάρια και τόνους αλλά παρουσιάζουν τα δεδομένα με βάσει την στήλη του τζίρου μόνο και δεν χρησιμοποιούν την στήλη Measure για τις γραφικές παραστάσεις μας.

8ο Κεφάλαιο – Τελικός κώδικας Project

Ο τελικός κώδικας μα είναι ο ακόλουθος :

```
import pandas as pd
import matplotlib.pyplot as plt
import mysql.connector
import csv
import tkinter as tk
from tkinter import messagebox
from tkinter import *

# ΔΙΑΒΑΣΤΕ ΤΟ CSV
# καθορίστε τη διεύθυνση URL του αρχείου CSV προς ανάγνωση
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'

# αποθήκευσε το csv σε ένα DataFrame
df = pd.read_csv(url)

# ================== ερώτημα 1ο ==========================
months_order = ['January', 'February', 'March', 'April', 'May', 'June', 'July',
'August', 'September', 'October', 'November', 'December']

# Δημιουργήστε μια νέα στήλη που περιέχει μόνο τις πληροφορίες του month από τη στήλη Date
df['Month'] = pd.to_datetime(df['Date'], format='%d/%m/%Y').dt.strftime('%B')

# Μετατρέψτε τη στήλη Month σε κατηγορηματική στήλη με την καθορισμένη σειρά
df['Month'] = pd.Categorical(df['Month'], categories=months_order, ordered=True)

# Ομαδοποιήστε ανά  month και αθροίστε τις τιμές
totals = df.groupby(['Month', 'Measure'])['Value'].sum().reset_index()

# Φιλτράρετε τα σύνολα USD και δημιουργήστε ένα DataFrame με στήλες Month και Value columns
totals_usd1 = totals[totals['Measure'] == '$'][['Month', 'Value']]

# Φιλτράρετε τα σύνολα Tonnes και δημιουργήστε ένα DataFrame με στήλες Month and Value columns
totals_ton1 = totals[totals['Measure'] == 'Tonnes'][['Month', 'Value']]


# ================== ερώτημα 2ο ==========================#
# ομαδοποιηση των στοιχείων που θελουμε
grouped = df.groupby(['Country', 'Measure'])
totals = grouped['Value'].sum()
```

```
# Φιλτράρετε τα σύνολα USD και δημιουργήστε ένα DataFrame με στήλες Country και Value
columns
totals_usd2 = totals.reset_index()
totals_usd2 = totals_usd2[totals_usd2['Measure'] == '$']
totals_usd2 = totals_usd2[['Country', 'Value']]

# Φιλτράρετε τα σύνολα Tonnes και δημιουργήστε ένα DataFrame με στήλες Country και
Value columns
totals_ton2 = totals.reset_index()
totals_ton2 = totals_ton2[totals_ton2['Measure'] == 'Tonnes']
totals_ton2 = totals_ton2[['Country', 'Value']]


# ================== ερώτημα 3ο ==========================#
# ομαδοποιηση των στοιχείων που θελουμε
grouped = df.groupby(['Transport_Mode', 'Measure'])
totals = grouped['Value'].sum()

# Φιλτράρετε τα σύνολα USD και δημιουργήστε ένα DataFrame με στήλες Transport_Mode και
Value columns
totals_usd3 = totals.reset_index()
totals_usd3 = totals_usd3[totals_usd3['Measure'] == '$']
totals_usd3 = totals_usd3[['Transport_Mode', 'Value']]

# Φιλτράρετε τα σύνολα Tonnes και δημιουργήστε ένα DataFrame με στήλες Transport_Mode
και Value columns
totals_ton3 = totals.reset_index()
totals_ton3 = totals_ton3[totals_ton3['Measure'] == 'Tonnes']
totals_ton3 = totals_ton3[['Transport_Mode', 'Value']]


# ================== ερώτημα 4ο ==========================#
# Καθορίστε τη σειρά των καθημερινών
weekday_order = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday',
'Saturday']

# Μετατρέψτε τη στήλη Weekday σε κατηγορηματικό τύπο δεδομένων με την καθορισμένη σειρά
df['Weekday'] = pd.Categorical(df['Weekday'], categories=weekday_order, ordered=True)

# ομαδοποιηση των στοιχείων που θελουμε
grouped = df.groupby(['Weekday', 'Measure'])
totals = grouped['Value'].sum()
# Φιλτράρετε τα σύνολα USD και δημιουργήστε ένα DataFrame με στήλες Weekday και Value
columns
totals_usd4 = totals.reset_index()
totals_usd4 = totals_usd4[totals_usd4['Measure'] == '$']
totals_usd4 = totals_usd4[['Weekday', 'Value']]
# Φιλτράρετε τα σύνολα Tonnes και δημιουργήστε ένα DataFrame με στήλες Weekday και
Value columns
totals_ton4 = totals.reset_index()
totals_ton4 = totals_ton4[totals_ton4['Measure'] == 'Tonnes']
totals_ton4 = totals_ton4[['Weekday', 'Value']]

# ================== ερώτημα 5ο ==========================#
# ομαδοποιηση των στοιχείων που θελουμε
grouped = df.groupby(['Commodity', 'Measure'])
totals = grouped['Value'].sum()

# Φιλτράρετε τα σύνολα USD και δημιουργήστε ένα DataFrame με στήλες Commodity και Value
columns
totals_usd5 = totals.reset_index()
totals_usd5 = totals_usd5[totals_usd5['Measure'] == '$']
totals_usd5 = totals_usd5[['Commodity', 'Value']]
# Φιλτράρετε τα σύνολα Tonnes και δημιουργήστε ένα DataFrame με στήλες Commodity και
```

```
Value columns
totals_ton5 = totals.reset_index()
totals_ton5 = totals_ton5[totals_ton5['Measure'] == 'Tonnes']
totals_ton5 = totals_ton5[['Commodity', 'Value']]

# ================== ερώτημα 6ο ==========================#

# Ομαδοποιήστε τα δεδομένα ανά month και υπολογίστε τον συνολικό τζιρο σε όλα τα μέσα και τους τύπους ανακυκλώσιμων
totals_all = df.groupby('Month')['Value'].sum().reset_index()

# Ταξινομήστε τα δεδομένα κατά τον συνολικό τζιρο με φθίνουσα σειρά
totals_all = totals_all.sort_values(by='Value', ascending=False)

# Επιλέξτε τους κορυφαίους 5 μήνες με τον υψηλότερο συνολικό τζιρο
top_5_months = totals_all.head(5)

# ================== ερώτημα 7ο ==========================#

# Ομαδοποιήστε τα δεδομένα ανά Country και Commodity και υπολογίστε τον συνολικό τζιρο σε όλα τα μέσα και τους τύπους ανακυκλώσιμων
totals_country_commodity = df.groupby(['Country',
'Commodity'])['Value'].sum().reset_index()

# Ταξινομήστε τα δεδομένα κατά τον συνολικό τζιρο με φθίνουσα σειρά
totals_country_commodity = totals_country_commodity.sort_values(by=['Country',
'Value'], ascending=[True, False])

# Επιλέξτε τισ 5 κορυφαιεσ κατηγορίες με τον υψηλότερο συνολικό τζιρο
top_commodity_country = totals_country_commodity.groupby('Country').head(5)

# ================== ερώτημα 8ο ========================== #

# Ομαδοποίηση δεδομένων βάσει Commodity και Date and και υπολογισμος του συνολικου τζιρου για καθε ομαδα
totals_commodity_date = df.groupby(['Commodity', 'Date'])['Value'].sum().reset_index()

# Ταξινομήστε τα δεδομένα κατά Commodity και Value σε φθίνουσα σειρά
totals_commodity_date = totals_commodity_date.sort_values(by=['Commodity', 'Value'],
ascending=[True, False])

# Επιλέξτε το κορυφαίο commodity για κάθε κατηγορία
top_commodity_category = totals_commodity_date.groupby('Commodity').head(1)

# ====================== Δεδομένα SQL ================================== #
#δημιουργια σύνδεσης με την SQL
db = mysql.connector.connect(
    host="localhost",
    user="root",
    passwd="karagiannis",
    database="Covid_19_Data"
)

mycursor = db.cursor()
# ====================== tables για ερώτημα 1 ================================== #
# δημιουργια πινάκων ερωτήματος 1
create_table_query1 = "CREATE TABLE Total_usd_per_month (Month VARCHAR(255), DOLLARS
FLOAT)"
mycursor.execute(create_table_query1)

create_table_query2 = "CREATE TABLE Total_ton_per_month (Month VARCHAR(255), TONNES
FLOAT)"
mycursor.execute(create_table_query2)

# εισαγωγή δεδομένων στους πίνακεσ του ερωτήματος 1
```

```
for _, row in totals_usd1.iterrows():
    insert_query1 = "INSERT INTO Total_usd_per_month (Month, DOLLARS) VALUES (%s, %s)"
    mycursor.execute(insert_query1, (row['Month'], float(row['Value'])))

for _, row in totals_ton1.iterrows():
    insert_query2 = "INSERT INTO Total_ton_per_month (Month, TONNES) VALUES (%s, %s)"
    mycursor.execute(insert_query2, (row['Month'], float(row['Value'])))

# ====================== tables για ερώτημα 2 ================================== #
# δημιουργια πινάκων ερωτήματος 2
create_table_query3 = "CREATE TABLE Total_usd_per_Country (Country VARCHAR(255),
DOLLARS FLOAT)"
mycursor.execute(create_table_query3)

create_table_query4 = "CREATE TABLE Total_ton_per_Country (Country VARCHAR(255), TONNES
FLOAT)"
mycursor.execute(create_table_query4)

# εισαγωγή δεδομένων στους πίνακεσ του ερωτήματος 2

for _, row in totals_usd2.iterrows():
    insert_query3 = "INSERT INTO Total_usd_per_Country (Country, DOLLARS) VALUES (%s,
%s)"
    mycursor.execute(insert_query3, (row['Country'], float(row['Value'])))

for _, row in totals_ton2.iterrows():
    insert_query4 = "INSERT INTO Total_ton_per_Country (Country, TONNES) VALUES (%s,
%s)"
    mycursor.execute(insert_query4, (row['Country'], float(row['Value'])))

# ====================== tables για ερώτημα 3 ================================== #
# δημιουργια πινάκων ερωτήματος 3
create_table_query5 = "CREATE TABLE Total_usd_per_Transport (Transport VARCHAR(255),
DOLLARS FLOAT)"
mycursor.execute(create_table_query5)

create_table_query6 = "CREATE TABLE Total_ton_per_Transport (Transport VARCHAR(255),
TONNES FLOAT)"
mycursor.execute(create_table_query6)

# εισαγωγή δεδομένων στους πίνακεσ του ερωτήματος 3

for _, row in totals_usd3.iterrows():
    insert_query5 = "INSERT INTO Total_usd_per_Transport (Transport, DOLLARS) VALUES
(%s, %s)"
    mycursor.execute(insert_query5, (row['Transport_Mode'], float(row['Value'])))

for _, row in totals_ton3.iterrows():
    insert_query6 = "INSERT INTO Total_ton_per_Transport (Transport, TONNES) VALUES
(%s, %s)"
    mycursor.execute(insert_query6, (row['Transport_Mode'], float(row['Value'])))

# ====================== tables για ερώτημα 4 ================================== #
# δημιουργια πινάκων ερωτήματος 4
create_table_query7 = "CREATE TABLE Total_usd_per_Weekday (Weekday VARCHAR(255),
DOLLARS FLOAT)"
mycursor.execute(create_table_query7)

create_table_query8 = "CREATE TABLE Total_ton_per_Weekday (Weekday VARCHAR(255), TONNES
FLOAT)"
mycursor.execute(create_table_query8)

# εισαγωγή δεδομένων στους πίνακεσ του ερωτήματος 4
for _, row in totals_usd4.iterrows():
    insert_query7 = "INSERT INTO Total_usd_per_Weekday (Weekday, DOLLARS) VALUES (%s,
```

```
%s)"
    mycursor.execute(insert_query7, (row['Weekday'], float(row['Value'])))

for _, row in totals_ton4.iterrows():
    insert_query8 = "INSERT INTO Total_ton_per_Weekday (Weekday, TONNES) VALUES (%s,
%s)"
    mycursor.execute(insert_query8, (row['Weekday'], float(row['Value'])))

# ====================== tables για ερώτημα 5 ================================== #
# δημιουργια πινάκων ερωτήματος 5
create_table_query9 = "CREATE TABLE Total_usd_per_Commodity (Commodity VARCHAR(255),
DOLLARS FLOAT)"
mycursor.execute(create_table_query9)

create_table_query10 = "CREATE TABLE Total_ton_per_Commodity (Commodity VARCHAR(255),
TONNES FLOAT)"
mycursor.execute(create_table_query10)

# εισαγωγή δεδομένων στους πίνακεσ του ερωτήματος 5

for _, row in totals_usd5.iterrows():
    insert_query9 = "INSERT INTO Total_usd_per_Commodity (Commodity, DOLLARS) VALUES
(%s, %s)"
    mycursor.execute(insert_query9, (row['Commodity'], float(row['Value'])))

for _, row in totals_ton5.iterrows():
    insert_query10 = "INSERT INTO Total_ton_per_Commodity (Commodity, TONNES) VALUES
(%s, %s)"
    mycursor.execute(insert_query10, (row['Commodity'], float(row['Value'])))

# ====================== tables για ερώτημα 6 ================================== #
# δημιουργια πινάκων ερωτήματος 6
create_table_query11 = "CREATE TABLE Top5_months (Month VARCHAR(255), Value FLOAT)"
mycursor.execute(create_table_query11)

# εισαγωγή δεδομένων στους πίνακεσ του ερωτήματος 6

for _, row in top_5_months.iterrows():
    insert_query11 = "INSERT INTO Top5_months (Month, Value) VALUES (%s, %s)"
    mycursor.execute(insert_query11, (row['Month'], float(row['Value'])))

# ====================== tables για ερώτημα 7 ================================== #
# δημιουργια πινάκων ερωτήματος 7
create_table_query12 = "CREATE TABLE Top5_Commodities (Country VARCHAR(255), Commodity
VARCHAR(255), Value FLOAT)"
mycursor.execute(create_table_query12)

# εισαγωγή δεδομένων στους πίνακεσ του ερωτήματος 7

for country in top_commodity_country['Country'].unique():
    data = top_commodity_country[top_commodity_country['Country'] == country]
    for _, row in data.iterrows():
        insert_query12 = "INSERT INTO Top5_Commodities (Country, Commodity, Value)
VALUES (%s, %s, %s)"
        mycursor.execute(insert_query12, (country, row['Commodity'],
float(row['Value'])))


# ====================== tables για ερώτημα 8 ================================== #
# δημιουργια πινάκων ερωτήματος 8
create_table_query13 = "CREATE TABLE Top1_DATE (Commodity VARCHAR(255), Date
VARCHAR(255), Value FLOAT)"
mycursor.execute(create_table_query13)

# εισαγωγή δεδομένων στους πίνακεσ του ερωτήματος 8
```

```
for commodity in top_commodity_category['Commodity'].unique():
    data = top_commodity_category[top_commodity_category['Commodity'] == commodity]
    for _, row in data.iterrows():
        insert_query13 = "INSERT INTO Top1_DATE (Commodity, Date, Value) VALUES (%s,
%s, %s)"
        mycursor.execute(insert_query13, (commodity, row['Date'], float(row['Value'])))

# ============================== select για ερώτημα 1 ================================
# Εξαγωγή δεδομένων ερωτήματος 1
select_query1 = "SELECT * FROM Total_usd_per_month"
mycursor.execute(select_query1)
data1 = mycursor.fetchall()

select_query2 = "SELECT * FROM Total_ton_per_month"
mycursor.execute(select_query2)
data2 = mycursor.fetchall()

# ============================== select για ερώτημα 2 ================================
# Εξαγωγή δεδομένων ερωτήματος 2
select_query3 = "SELECT * FROM Total_usd_per_Country"
mycursor.execute(select_query3)
data3 = mycursor.fetchall()

select_query4 = "SELECT * FROM Total_ton_per_Country"
mycursor.execute(select_query4)
data4 = mycursor.fetchall()
# ============================== select για ερώτημα 3 ================================
# Εξαγωγή δεδομένων ερωτήματος 3
select_query5 = "SELECT * FROM Total_usd_per_Transport"
mycursor.execute(select_query5)
data5 = mycursor.fetchall()

select_query6 = "SELECT * FROM Total_ton_per_Transport"
mycursor.execute(select_query6)
data6 = mycursor.fetchall()
# ============================== select για ερώτημα 4 ================================
# Εξαγωγή δεδομένων ερωτήματος 4
select_query7 = "SELECT * FROM Total_usd_per_Weekday"
mycursor.execute(select_query7)
data7 = mycursor.fetchall()

select_query8 = "SELECT * FROM Total_ton_per_Weekday"
mycursor.execute(select_query8)
data8 = mycursor.fetchall()
# ============================== select για ερώτημα 5 ================================
# Εξαγωγή δεδομένων ερωτήματος 5
select_query9 = "SELECT * FROM Total_usd_per_Commodity"
mycursor.execute(select_query9)
data9 = mycursor.fetchall()

select_query10 = "SELECT * FROM Total_ton_per_Commodity"
mycursor.execute(select_query10)
data10 = mycursor.fetchall()
# ============================== select για ερώτημα 6 ================================
# Εξαγωγή δεδομένων ερωτήματος 6
select_query11 = "SELECT * FROM Top5_months"
mycursor.execute(select_query11)
data11 = mycursor.fetchall()
# ============================== select για ερώτημα 7 ================================
# Εξαγωγή δεδομένων ερωτήματος 7
select_query12 = "SELECT * FROM Top5_Commodities"
mycursor.execute(select_query12)
data12 = mycursor.fetchall()
# ============================== select για ερώτημα 8 ================================
# Εξαγωγή δεδομένων ερωτήματος 8
```

```
select_query13 = "SELECT * FROM Top1_DATE"
mycursor.execute(select_query13)
data13 = mycursor.fetchall()

# Τερματισμός ένωσης με την Βάση
db.commit()
db.close()
# ============================ CSV ==============================
# Δημιοργεία αρχείου CSV
csv_file = 'Covid_19_Data.csv'

# ============================ write the datas to csv ============================
# Εγγραφή δεδομένων στο CSV
with open(csv_file, 'w', newline='') as file:
    writer = csv.writer(file)

    # ============================ pinakas 1 ============================

    # Δημιουργία κεφαλής για τον πίνακα 1
    writer.writerow(['Month', 'Dollars'])

    # Εισαγωγή δεδομένων πίνακα 1
    for row1 in data1:
        writer.writerow(row1)

    # Δημιουργία κενής κεφαλής
    writer.writerow([])

    # Δημιουργία κεφαλής για τον πίνακα 2
    writer.writerow(['Month', 'Tonnes'])

    # Εισαγωγή δεδομένων πίνακα 2
    for row2 in data2:
        writer.writerow(row2)

    # Δημιουργία κενής κεφαλής
    writer.writerow([])

    # ============================ pinakas 2 ============================

    # Δημιουργία κεφαλής για τον πίνακα 3
    writer.writerow(['Country', 'Dollars'])

    # Εισαγωγή δεδομένων πίνακα 3
    for row3 in data3:
        writer.writerow(row3)

    # Δημιουργία κενής κεφαλής
    writer.writerow([])

    # Δημιουργία κεφαλής για τον πίνακα 4
    writer.writerow(['Country', 'Tonnes'])

    # Εισαγωγή δεδομένων πίνακα 4'
    for row4 in data4:
        writer.writerow(row4)

    # Δημιουργία κενής κεφαλής
    writer.writerow([])

    # ============================ pinakas 3 ============================

    # Δημιουργία κεφαλής για τον πίνακα 5
    writer.writerow(['Transport', 'Dollars'])
```

```
    # Εισαγωγή δεδομένων πίνακα 5
    for row5 in data5:
        writer.writerow(row5)

    # Δημιουργία κενής κεφαλής
    writer.writerow([])

    # Δημιουργία κεφαλής για τον πίνακα 6
    writer.writerow(['Transport', 'Tonnes'])

    # Εισαγωγή δεδομένων πίνακα 5
    for row6 in data6:
        writer.writerow(row6)

    # Δημιουργία κενής κεφαλής
    writer.writerow([])

    # ============================== pinakas 4 ==============================

    # Δημιουργία κεφαλής για τον πίνακα 7
    writer.writerow(['Weekday', 'Dollars'])

    # Εισαγωγή δεδομένων πίνακα 7
    for row7 in data7:
        writer.writerow(row7)

    # Δημιουργία κενής κεφαλής
    writer.writerow([])

    # Δημιουργία κεφαλής για τον πίνακα 8
    writer.writerow(['Weekday', 'Tonnes'])

    # Εισαγωγή δεδομένων πίνακα 8
    for row8 in data8:
        writer.writerow(row8)

    # Δημιουργία κενής κεφαλής
    writer.writerow([])

    # ============================== pinakas 5 ==============================

    # Δημιουργία κεφαλής για τον πίνακα 9
    writer.writerow(['Commodity', 'Dollars'])

    # Εισαγωγή δεδομένων πίνακα 9
    for row9 in data9:
        writer.writerow(row9)

    # Δημιουργία κενής κεφαλής
    writer.writerow([])

    # Δημιουργία κεφαλής για τον πίνακα 10
    writer.writerow(['Commodity', 'Tonnes'])

    # Εισαγωγή δεδομένων πίνακα 10
    for row10 in data10:
        writer.writerow(row10)

    # Δημιουργία κενής κεφαλής
    writer.writerow([])

    # ============================== pinakas 6 ==============================

    # Δημιουργία κεφαλής για τον πίνακα 11
    writer.writerow(['Month', 'Dollars'])
```

```
    # Εισαγωγή δεδομένων πίνακα 11
    for row11 in data11:
        writer.writerow(row11)

    # Δημιουργία κενής κεφαλής
    writer.writerow([])

    # ============================ pinakas 7 ==============================

    # Δημιουργία κεφαλής για τον πίνακα 12
    writer.writerow(['Country', 'Commodity', 'Value'])

    # Εισαγωγή δεδομένων πίνακα 12
    for row12 in data12:
        writer.writerow(row12)

    # Δημιουργία κενής κεφαλής
    writer.writerow([])

    # ============================ pinakas 8 ==============================

    # Δημιουργία κεφαλής για τον πίνακα 13
    writer.writerow(['Commodity', 'Date', 'Value'])

    # Εισαγωγή δεδομένων πίνακα 13
    for row13 in data13:
        writer.writerow(row13)

# δημιουργία συνάρτησης για το ερώτημα 1


def show_chart1():
    fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

    totals_usd1.plot(kind='bar', x='Month', y='Value', ax=ax1, color='grey')
    ax1.set_title('Total Turnover in USD by Month')
    ax1.set_xlabel('Month')
    ax1.set_ylabel('Total Turnover (USD)')

    totals_ton1.plot(kind='bar', x='Month', y='Value', ax=ax2, color='maroon')
    ax2.set_title('Total Turnover in Tonnes by Month')
    ax2.set_xlabel('Month')
    ax2.set_ylabel('Total Turnover (Tonnes)')

    plt.show()

# δημιουργία συνάρτησης για το ερώτημα 2


def show_chart2():
    fig, (ax3, ax4) = plt.subplots(1, 2, figsize=(12, 6))

    totals_usd2.plot(kind='bar', x='Country', y='Value', ax=ax3, color='grey')
    ax3.set_title('Total Turnover in USD by Country')
    ax3.set_xlabel('Country')
    ax3.set_ylabel('Total Turnover (USD)')

    totals_ton2.plot(kind='bar', x='Country', y='Value', ax=ax4, color='maroon')
    ax4.set_title('Total Turnover in Tonnes by Country')
    ax4.set_xlabel('Country')
    ax4.set_ylabel('Total Turnover (Tonnes)')

    plt.show()
```

```
# δημιουργία συνάρτησης για το ερώτημα 3


def show_chart3():
    fig, (ax5, ax6) = plt.subplots(1, 2, figsize=(12, 6))

    totals_usd3.plot(kind='bar', x='Transport_Mode', y='Value', ax=ax5, color='grey')
    ax5.set_title('Total Turnover in USD by Transport_Mode')
    ax5.set_xlabel('Transport_Mode')
    ax5.set_ylabel('Total Turnover (USD)')

    totals_ton3.plot(kind='bar', x='Transport_Mode', y='Value', ax=ax6, color='maroon')
    ax6.set_title('Total Turnover in Tonnes by Transport_Mode')
    ax6.set_xlabel('Transport_Mode')
    ax6.set_ylabel('Total Turnover (Tonnes)')

    plt.show()

# δημιουργία συνάρτησης για το ερώτημα 4


def show_chart4():
    fig, (ax7, ax8) = plt.subplots(1, 2, figsize=(12, 6))

    totals_usd4.plot(kind='bar', x='Weekday', y='Value', ax=ax7, color='grey')
    ax7.set_title('Total Turnover in USD by Weekday')
    ax7.set_xlabel('Weekday')
    ax7.set_ylabel('Total Turnover (USD)')

    totals_ton4.plot(kind='bar', x='Weekday', y='Value', ax=ax8, color='maroon')
    ax8.set_title('Total Turnover in Tonnes by Weekday')
    ax8.set_xlabel('Weekday')
    ax8.set_ylabel('Total Turnover (Tonnes)')

    plt.show()

# δημιουργία συνάρτησης για το ερώτημα 5


def show_chart5():
    fig, (ax9, ax10) = plt.subplots(1, 2, figsize=(12, 6))

    totals_usd5.plot(kind='bar', x='Commodity', y='Value', ax=ax9, color='grey')
    ax9.set_title('Total Turnover in USD by Commodity')
    ax9.set_xlabel('Commodity')
    ax9.set_ylabel('Total Turnover (USD)')

    totals_ton5.plot(kind='bar', x='Commodity', y='Value', ax=ax10, color='maroon')
    ax10.set_title('Total Turnover in Tonnes by Commodity')
    ax10.set_xlabel('Commodity')
    ax10.set_ylabel('Total Turnover (Tonnes)')

    plt.show()

# δημιουργία συνάρτησης για το ερώτημα 6


def show_chart6():
    explode = (0.1, 0.0, 0.0, 0.0, 0.0)
    # Create a pie chart to visualize the data
    fig, ax11 = plt.subplots(figsize=(8, 6))

    top_5_months.plot(kind='pie', y='Value', labels=top_5_months['Month'], ax=ax11,
                      colors=['blue', 'green', 'yellow', 'silver', 'red'], shadow=True,
startangle=90, explode=explode,
```

```
                      autopct='%1.1f%%')
    ax11.set_title('Top 5 Months with the Highest Turnover')
    ax11.set_ylabel('')

    plt.show()

# δημιουργία συνάρτησης για το ερώτημα 7


def show_chart7():
    for country in top_commodity_country['Country'].unique():
        data = top_commodity_country[top_commodity_country['Country'] == country]

        # Create the bar chart
        fig, ax12 = plt.subplots(figsize=(8, 6))
        ax12.bar(data['Commodity'], data['Value'], width=0.2)
        ax12.set_title('Top 5 Commodities for {}'.format(country))
        ax12.set_xlabel('Commodity')
        ax12.set_ylabel('Total Turnover')

        # Rotate the x-axis labels for better readability
        plt.xticks(rotation=45)

        plt.show()

# δημιουργία συνάρτησης για το ερώτημα 8


def show_chart8():
    for commodity in top_commodity_category['Commodity']:
        data = top_commodity_category[top_commodity_category['Commodity'] == commodity]

        # Create the bar chart
        fig, ax13 = plt.subplots(figsize=(8, 6))
        ax13.bar(data['Date'], data['Value'], width=0.2)
        ax13.set_title('Top Date for {}'.format(commodity))
        ax13.set_xlabel('Date')
        ax13.set_ylabel('Total Turnover')

        # Rotate the x-axis labels for better readability
        plt.xticks(rotation=0)

        # Show the plot
        plt.show()

# δημιουργία δεύτερου παραθύρου για το Gui


def open_new_window():
    def go_back():
        new_window.destroy()
        root.deiconify()
    new_window = Toplevel(root)
    new_window.geometry("800x600")
    new_window.title("Charts")
    new_window.configure(background='#E0E0EE')
    # δημιοργία label παραθύρου
    Label(new_window, text="Καλώς ήρθατε στην ενότητα των γραφημάτων!", font="Times
22", padx=20, pady=10, background='#E0E0EE').pack()
    Label(new_window, text="Σε αυτήν την ενότητα μπορείτε να δείτε και να
επεξεργαστείτε τα "
                           "γραφήματα που προέκυψαν έπειτα\n από σχολαστική
μελέτη του αρχείου που μας ζητήθηκε."
                           "Παρακάτω σας παραθέτω τους τίτλους\n του κάθε
γραφήματος για την περεταίρω διευκόλυνση σας."
```

```
          , font="Times 14", padx=30, pady=10, background='#E0E0EE').pack()
    Label(new_window, text="Chart1:Συνολική παρουσίαση του τζίρου ανά μήνα.\n"
                           "Chart2:Συνολική παρουσίαση του τζίρου για κάθε χώρα.\n"
                           "Chart3:Συνολική παρουσίαση του τζίρου για κάθε μέσο
μεταφοράς.\n"
                           "Chart4:Συνολική παρουσίαση του τζίρου για κάθε μέρα της
εβδομάδας.\n"
                           "Chart5:Συνολική παρουσίαση του τζίρου για κάθε κατηγορία
εμπορεύματος.\n"
                           "Chart6:Παρουσίαση των 5 μηνών με το μεγαλύτερο
τζίρο,ανεξαρτήτως\n μέσου μεταφοράς και είδους ανακυκλώσιμων ειδών.\n"
                           "Chart7:Παρουσίαση των 5 κατηγοριών εμπορευμάτων με το
μεγαλύτερο τζίρο,για κάθε χώρα.\n"
                           "Chart8:Παρουσίαση της ημέρας με το μεγαλύτερο τζίρο, για
κάθε κατηγορία εμπορεύματος.\n"
          , font="Times 14", padx=25, pady=10, background='#E0E0EE').pack()
    # δημιοργία drop down κουμπιου
    chart_options = ["Chart 1", "Chart 2", "Chart 3", "Chart 4", "Chart 5", "Chart 6",
"Chart 7", "Chart 8"]
    selected_chart = tk.StringVar()
    selected_chart.set(chart_options[0])  # Set the default selected option

    chart_dropdown = OptionMenu(new_window, selected_chart, *chart_options)
    chart_dropdown.pack(padx=20, pady=20)

    def handle_selection():
        selected_option = selected_chart.get()
        if selected_option == "Chart 1":
            show_chart1()
        elif selected_option == "Chart 2":
            show_chart2()
        elif selected_option == "Chart 3":
            show_chart3()
        elif selected_option == "Chart 4":
            show_chart4()
        elif selected_option == "Chart 5":
            show_chart5()
        elif selected_option == "Chart 6":
            show_chart6()
        elif selected_option == "Chart 7":
            show_chart7()
        elif selected_option == "Chart 8":
            show_chart8()

        # δημιουργία κουμπιου για την εμφάνιση των γραφικών

    chart_button = tk.Button(new_window, text="Show Chart", command=handle_selection,
background='#EEEEE9')
    chart_button.pack(padx=20, pady=20)

    # δημιουργια κουμπιου που σε πάει πίσω
    back_button = tk.Button(new_window, text="Back", command=go_back,
background='#EEEEE9')
    back_button.pack(padx=20, pady=20)


def close_window():
    if messagebox.askokcancel("Close", "Are you sure you want to close the window?"):
        root.destroy()  # Close the current window

# δημιουργία πρώτου παραθύρου για το Gui
root = tk.Tk()
root.geometry("1000x400")
root.title("Covid_19 Data")
root.configure(background='#E0E0EE')
```

```
# δημιουργία label παραθύρου
Label(root, text="Covid-19 X Python ?", font="Times 22", padx=20, pady=10,
background='#E0E0EE').pack()
Label(root, text="Στο project αυτό, αναλύσαμε ένα αρχείο CSV για την επίδραση του
COVID-19 στο εμπόριο.\n"
                 "Χρησιμοποιώντας κατάλληλα ορίσαματα της γλώσσας Python,καταφέραμε να
διαβάσαμε τα δεδομένα\n"
                 " και δημιουργήσαμε τα απαραιτητα γραφήματα για την οπτική
αναπαράσταση τους.\n"
                 "Επειτα χρησιμοποιώντασ το περιβάλλων τησ Mysql, καταφέραμε να
αποθηκεύσουμε τα δεδομένα \n"
                 " σε πίνακες για την καλύτερη διαχείρηση τους για περεταίρω εργασίες
στο αμμεσο μέλλον.\n"
                 "Τα γραφήματα οπωσ θα δείται και εσείς μας αποκάλυψαν μια εικόνα της
επίδρασης του COVID-19 στο παγκόσμιο εμπόριο.\n"
                 "Εάν θέλετε να δείτε τα γραφήματα που πρόεκυψαν παρακαλώ πατείστε το
κουμπι Continue, αλλιώς μπορείτε να βγείτε \n"
                 "από την εφαρμογή πατώντας Exit. Σας  ευχαριστώ πολύ!"
          , font="Times 14", padx=25, pady=10, background='#E0E0EE').pack()
# δημιουργία κουμπιού συνέχισης
open_button = tk.Button(root, text="Continue", command=open_new_window,
background='#E0E0EE')
open_button.pack(padx=50, pady=20)

# δημιουργία κουμπιού τερματισμού
close_button = tk.Button(root, text="Exit", command=close_window, background='#E0E0EE')
close_button.pack(padx=0, pady=20)


root.mainloop()
```