Αναφορά Project Python

Μάθημα:
Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών

ΔΑΜΙΑΝΟΣ ΔΙΑΣΑΚΟΣ 1084632

Περιεχόμενα

1. Συγκέντρωση δεδομένων από το διαδίκτυο. 2
2. Χρήση της βιβλιοθήκης tkinter για την δημιουργία gui. 2
3. Κώδικας Ερωτημάτων. 5

3.1. Συνολική παρουσίαση του τζίρου (στήλη value) ανά μήνα (στις αντίστοιχες μονάδες μέτρησης) 5

3.2. Συνολική παρουσίαση του τζίρου (στήλη value) για κάθε χώρα (στις αντίστοιχες μονάδες μέτρησης) 7

3.3. Συνολική παρουσίαση του τζίρου (στήλη value) για κάθε μέσο μεταφοράς (στις αντίστοιχες μονάδες μέτρησης) 8

3.4. Συνολική παρουσίαση του τζίρου (στήλη value) για κάθε μέρα της εβδομάδας (στις αντίστοιχες μονάδες μέτρησης) 9

3.5. Συνολική παρουσίαση του τζίρου (στήλη value) για κάθε κατηγορία εμπορεύματος (στις αντίστοιχες μονάδες μέτρησης) 9

3.6. Παρουσίαση των 5 μηνών με το μεγαλύτερο τζίρο, ανεξαρτήτως μέσου μεταφοράς και είδους ανακυκλώσιμων ειδών 10

3.7. Παρουσίαση των 5 κατηγοριών εμπορευμάτων με το μεγαλύτερο τζίρο, για κάθε χώρα 11

3.8. Παρουσίαση της ημέρας με το μεγαλύτερο τζίρο, για κάθε κατηγορία εμπορεύματος 14

4. Σύνδεση με Βάση Δεδομένων 15
5. Παραδοχές 16

1. Συγκέντρωση δεδομένων από το διαδίκτυο.

```
# we use the url value to store the url of our desired dataset

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'


# In order to download the file we will need to send a get request

get_success = requests.get(url)


# If the request is successful the status  HTTP 200 OK will be returned as 200 that will mean that the request has succeeded.

if get_success.status_code == 200:


    # Save the dataset to a csv file

    with open('covid19.csv', 'wb') as csvfile:

        csvfile.write(get_success.content)


    print('File downloaded and saved successfully.')

else:

    # If the request didn't succeed

    print(f'Request failed!')


# Load the csv into a pandas dataframe in order to extract information from it and filter it.

df = pd.read_csv('covid19.csv')
```

2. Χρήση της βιβλιοθήκης tkinter για την δημιουργία gui.

Για την προσθήκη ενός **GUI** όπου ο χρήστης θα μπορεί να επιλέγει ποια πράξη θέλει να εκτελέσει θα χρησιμοποιηθεί η βιβλιοθήκη **tkinter.** Άρχικά, θα χρειαστεί να δημιουργηθεί ένα παράθυρο της **tinker** όπου θα ονομαστεί **root**. Έπειτα θα χρειαστεί να προσθέσουμε ορισμένα κουμπιά (**buttons**) έτσι ώστε η χρήση του **GUI** να είναι πιο εύκολη για τον χρήστη. Θα χρειαστεί

όμως και μία αναδιάταξη στις διαστάσεις των κουμπιών έτσι ώστε τα κουμπιά να είναι όσο γίνεται πιο ωραία διαμορφωμένα στο παράθυρο.

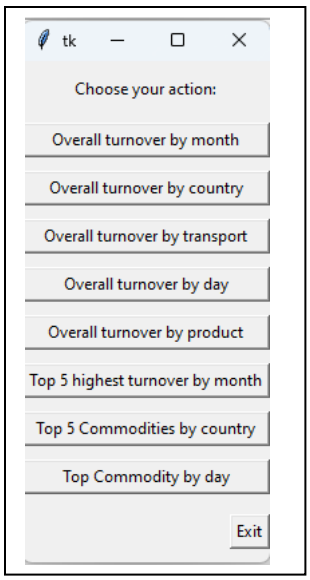
```
# Creating the tkinter window

root = tk.Tk()


# Creating the text label

label = tk.Label(root, text="Choose your action:")


# Create the buttons

button1 = tk.Button(root, text="Overall turnover by month", command=case_1)

button2 = tk.Button(root, text="Overall turnover by country", command=case_2)

button3 = tk.Button(root, text="Overall turnover by transport", command=case_3)

button4 = tk.Button(root, text="Overall turnover by day", command=case_4)

button5 = tk.Button(root, text="Overall turnover by product", command=case_5)

button6 = tk.Button(root, text="Top 5 highest turnover by month", command=case_6)

button7 = tk.Button(root, text="Top 5 Commodities by country", command=case_7)

button8 = tk.Button(root, text="Top Commodity by day'", command=case_8)


# Creating the exit button

exit_button = tk.Button(root, text="Exit", command=exit_program)
```

Για την διαμόρφωση των κουμπιών θα χρησιμοποιήσουμε

την εντολή **button1.config(width=button_width)** για κάθε κουμπί. Έπειτα θα τα προσθέσουμε στο «πλέγμα» του παραθύρου με την εντολή **button1.grid(row=1, column=1, sticky="e", pady=5)** για κάθε κουμπί. Το **row** και **column** και **sticky** συμβολίζουν που στον παράθυρο θα τοποθετηθεί το κουμπί. Συγκεκριμένα, το row και το column συμβολίζουν την γραμμή και την στήλη στο παράθυρο ενώ το sticky καθορίζει την ευθυγράμμιση του κουμπιού με το παράθυρο το οποίο μπορεί να πάρει τιμές όπως **e**,**s**,**n**,**w**,**se**,**sw** κα που συμβολίζουν την ανατολικά, νότια, βοριά, δυτικά, νότιο-ανατολικά και νότιο- δυτικά αντίστοιχα. Επίσης, το κουμπί της εξόδου θα το τοποθετηθεί νότιο- δυτικά για να είναι δείχνει λίγο πιο αισθητικά στο παράθυρο.

```
# we need to set the width of all the buttons to a desired width in order for them to fit
button_width = 25
button1.config(width=button_width)
button2.config(width=button_width)
button3.config(width=button_width)
button4.config(width=button_width)
button5.config(width=button_width)
button6.config(width=button_width)
button7.config(width=button_width)
button8.config(width=button_width)


# Add the buttons to the grid of our window
label.grid(row=0, column=0, columnspan=2, pady=10)
button1.grid(row=1, column=1, sticky="e", pady=5)
button2.grid(row=2, column=1, sticky="e", pady=5)
button3.grid(row=3, column=1, sticky="e", pady=5)
button4.grid(row=4, column=1, sticky="e", pady=5)
button5.grid(row=5, column=1, sticky="e", pady=5)
button6.grid(row=6, column=1, sticky="e", pady=5)
button7.grid(row=7, column=1, sticky="e", pady=5)
button8.grid(row=8, column=1, sticky="e", pady=5)
exit_button.grid(row=9, column=1, sticky="se", pady=10)
# Run the window
root.mainloop()
```

Στην συνέχεια, για να μπορέσουμε να εκτελέσουμε κάθε ερώτημα ξεχωριστά θα γίνει χρήση των **case** που ορίστηκαν κατά την δημιουργία τον κουμπιών. Όταν ένα κουμπί πατιέται θα εκτελείται και η συγκεκριμένη **case**, οι οποίες είναι δηλωμένες στον κώδικα με την μορφή **def case_1():** . Παρακάτω στο *Σχήμα 1*. απεικονίζεται το **Tkinter GUI** μαζί με τα κουμπιά.



Σχήμα 1. Gui χρησιμοποιώντας Tkinter

3. Κώδικας Ερωτημάτων.

3.1. Συνολική παρουσίαση του τζίρου (στήλη value) ανά μήνα (στις αντίστοιχες μονάδες μέτρησης)

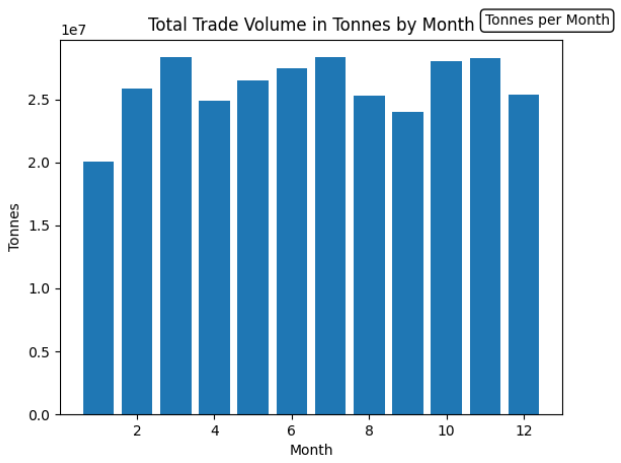
Αρχικά, η εντολή **df['Month'] = pd.to_datetime(df['Date'], dayfirst=True).dt.month** δημιουργεί μια στήλη «**Month**» στο **DataFrame df** και χρησιμοποιεί τη συνάρτηση **pd.to_datetime()** για να μετατρέψει τις τιμές της στήλης «**Date**» σε αντικείμενα **datetime**. Η τιμή **dayfirst=True** καθορίζει ότι οι ημερομηνίες έχουν τη μορφή «ηη/μμ/εεεε».

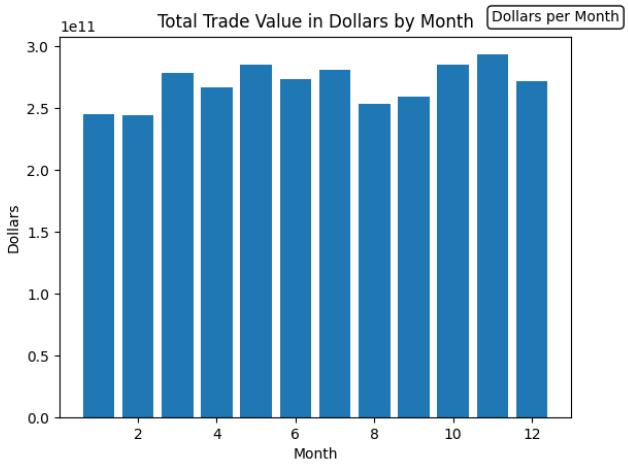
Στη συνέχεια, η εντολή **df_month = df.groupby(['Month', 'Measure'])['Value'].sum().reset_index()** Ομαδοποιεί το **DataFrame df** με βάση τις στήλες «**Month**» και «**Measure**» όπου έπειτα υπολογίζει το άθροισμα των τιμών του κάθε **Measure**. Στη συνέχεια θα χρειαστεί να εξάγουμε την πληροφορία μόνο για το «**Measure**» **$** για κάθε «**Month**» τα οποία τα προσθέτουμε σε μία λίστα ονόματι **dollar_months**.

Ακόμη, θα χρειαστεί να εκτυπώσουμε την λίστα για να φανούν τα δεδομένα μας. Το ίδιο θα κάνουμε και για την στήλη «**Measure**» **Tonnes**.

Στο τέλος, θα χρειαστεί να δημιουργηθούν γράφοι για τα αποτελέσματα που διεξήγαμε από το **dataset**. Για να κάνουμε plot τα δεδομένα θα χρησιμοποιήσουμε την βιβλιοθήκη **matplotlib.pyplot** όπου θα εμφανίσουμε 2 ξεχωριστά γραφήματα, το ένα για **$** και το άλλο για **Tonnes**. Χρειάστηκε επίσης να διαμορφωθεί το **text** στα γραφήματα διότι στον **x** άξονα τα ονόματα των μηνών επικαλύπταν το ένα το άλλο.

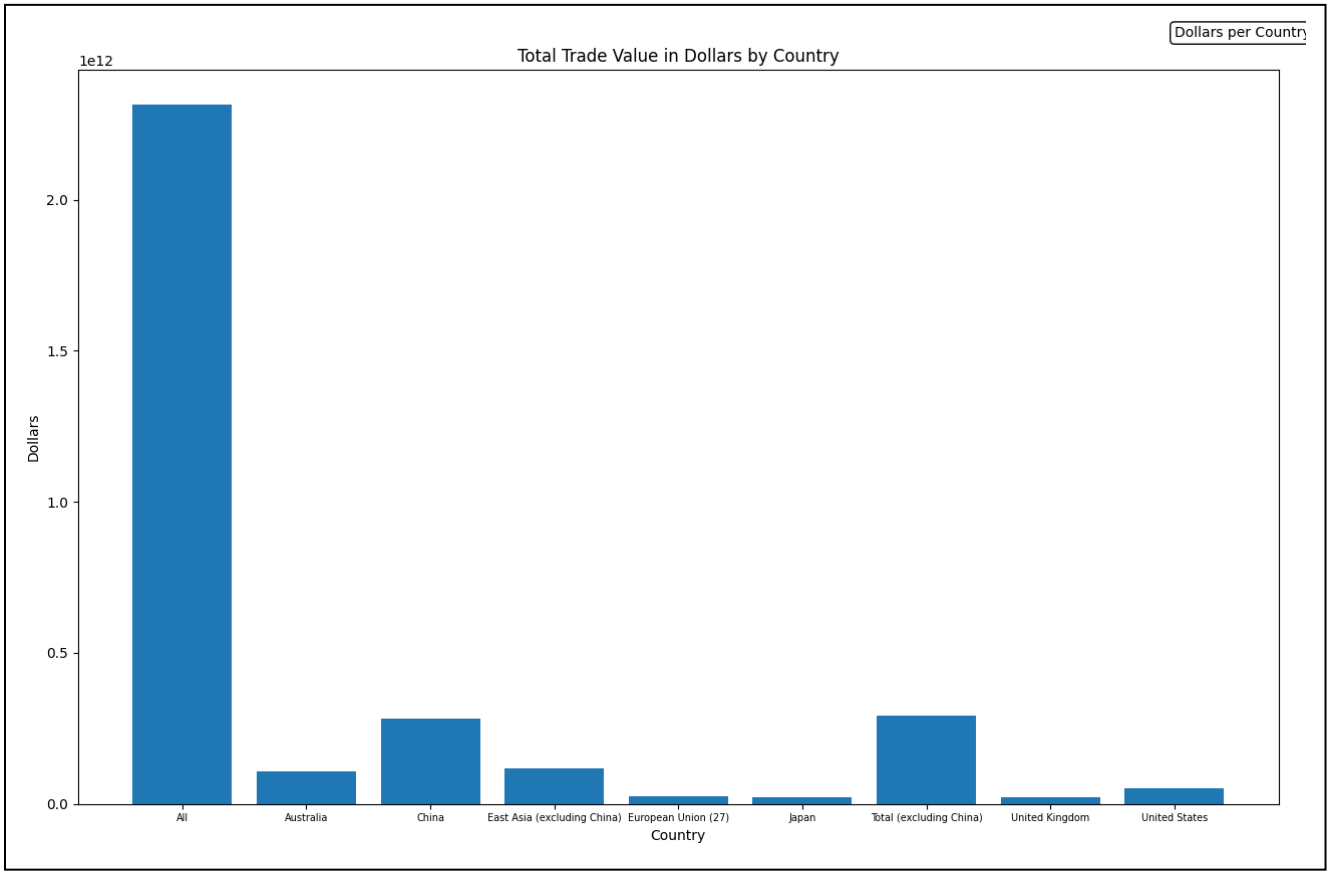
```
# Bar graph for dollars
    plt.bar(df_month[df_month['Measure'] == '$']['Month'], dollar_months)
    plt.xlabel('Month')
    plt.ylabel('Dollars')
    plt.title('Total Trade Value in Dollars by Month')

    # Add label to the top of the graph
    plt.text(0.97, 1.05, 'Dollars per Month', ha='center', va='center', transform=plt.gca().transAxes,
             bbox=dict(facecolor='white', edgecolor='black', boxstyle='round,pad=0.3'))

    plt.show()
    # Bar graph for tonnes
    plt.bar(df_month[df_month['Measure'] == 'Tonnes']['Month'], tonnes_months)
    plt.xlabel('Month')
    plt.ylabel('Tonnes')
    plt.title('Total Trade Volume in Tonnes by Month')

    # Add label to the top of the graph
    plt.text(0.97, 1.05, 'Tonnes per Month', ha='center', va='center', transform=plt.gca().transAxes,
             bbox=dict(facecolor='white', edgecolor='black', boxstyle='round,pad=0.3'))
    plt.show()
```

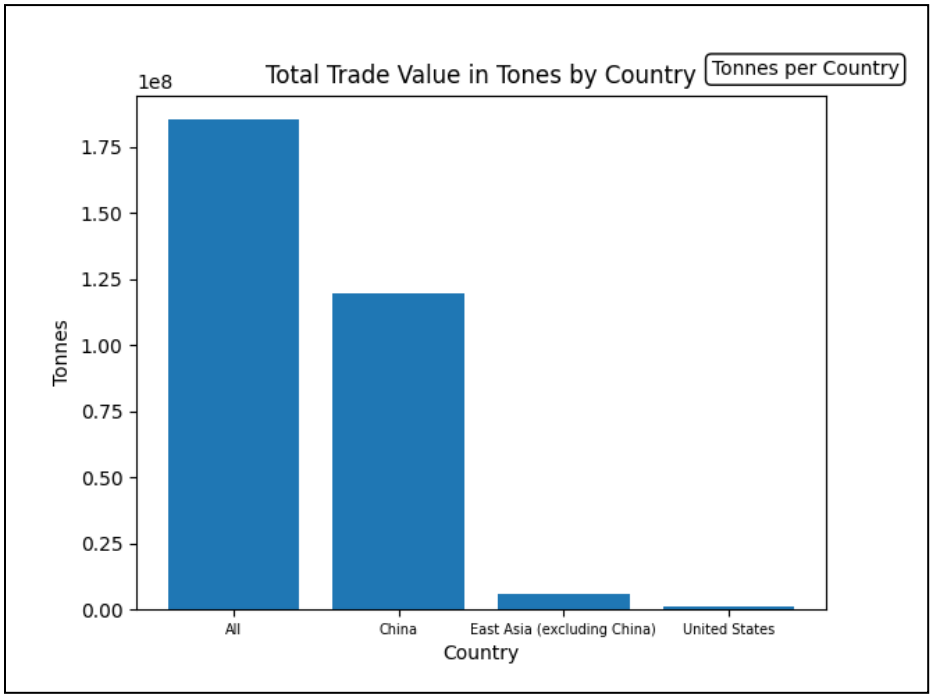




3.2. Συνολική παρουσίαση του τζίρου (στήλη value) για κάθε χώρα (στις αντίστοιχες μονάδες μέτρησης)

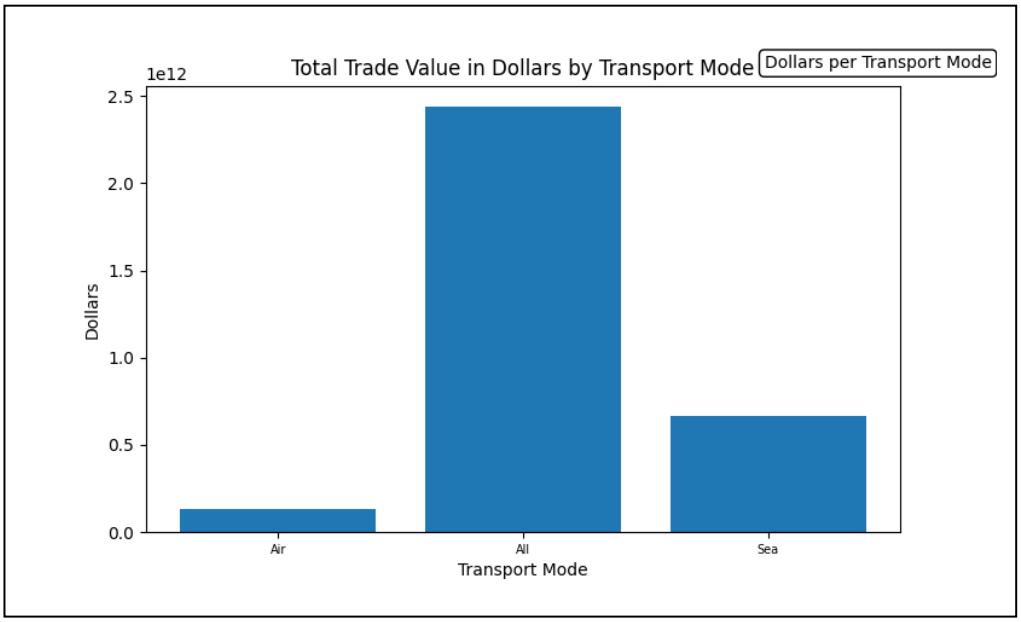
Ομοίως με το ερώτημα 3.1 κατασκευάσαμε τα παρακάτω γραφήματα:

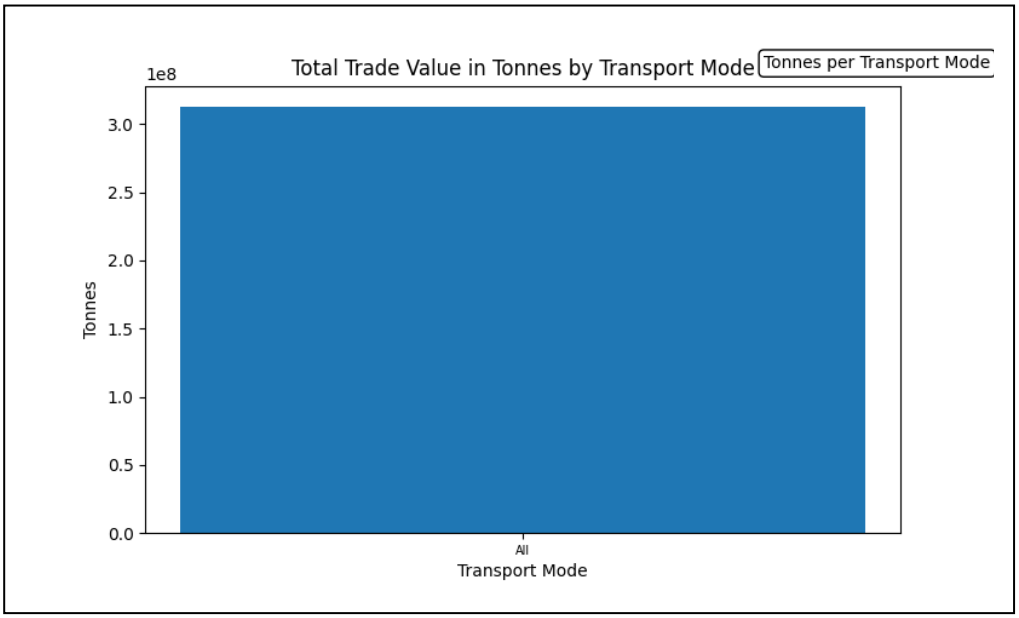




3.3. Συνολική παρουσίαση του τζίρου (στήλη value) για κάθε μέσο μεταφοράς (στις αντίστοιχες μονάδες μέτρησης)

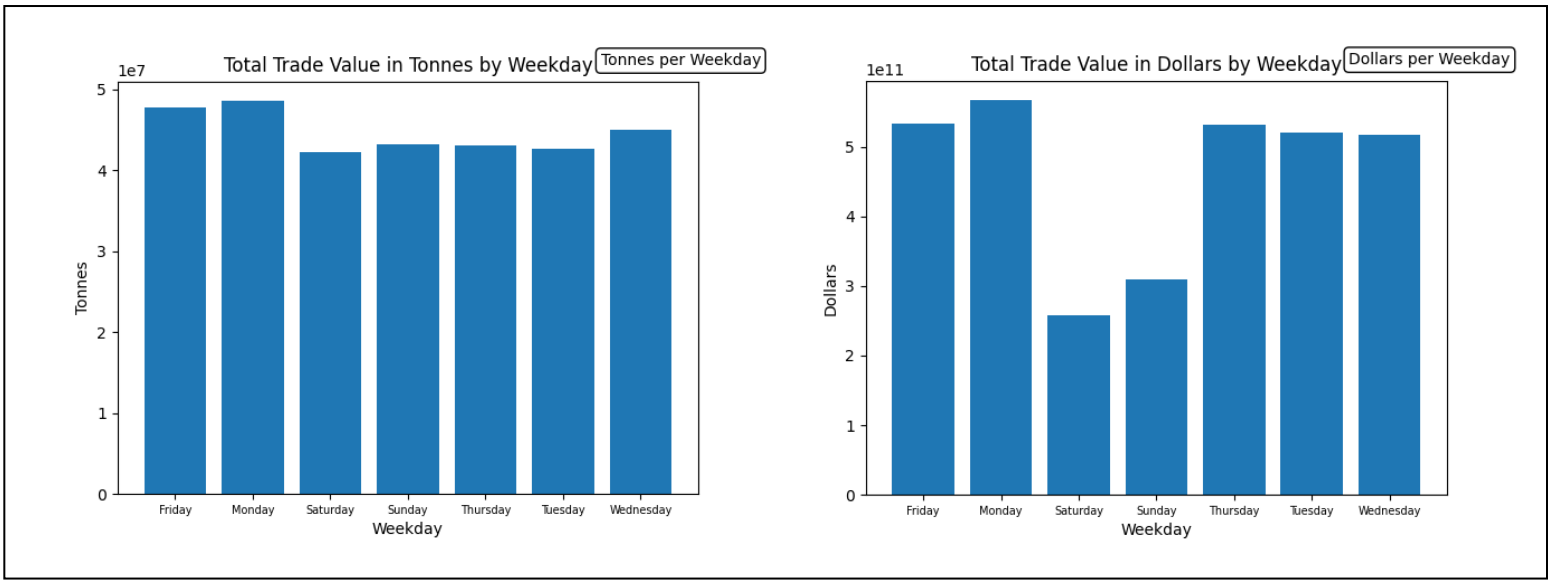
Ομοίως με το ερώτημα 3.1 κατασκευάσαμε τα παρακάτω γραφήματα:





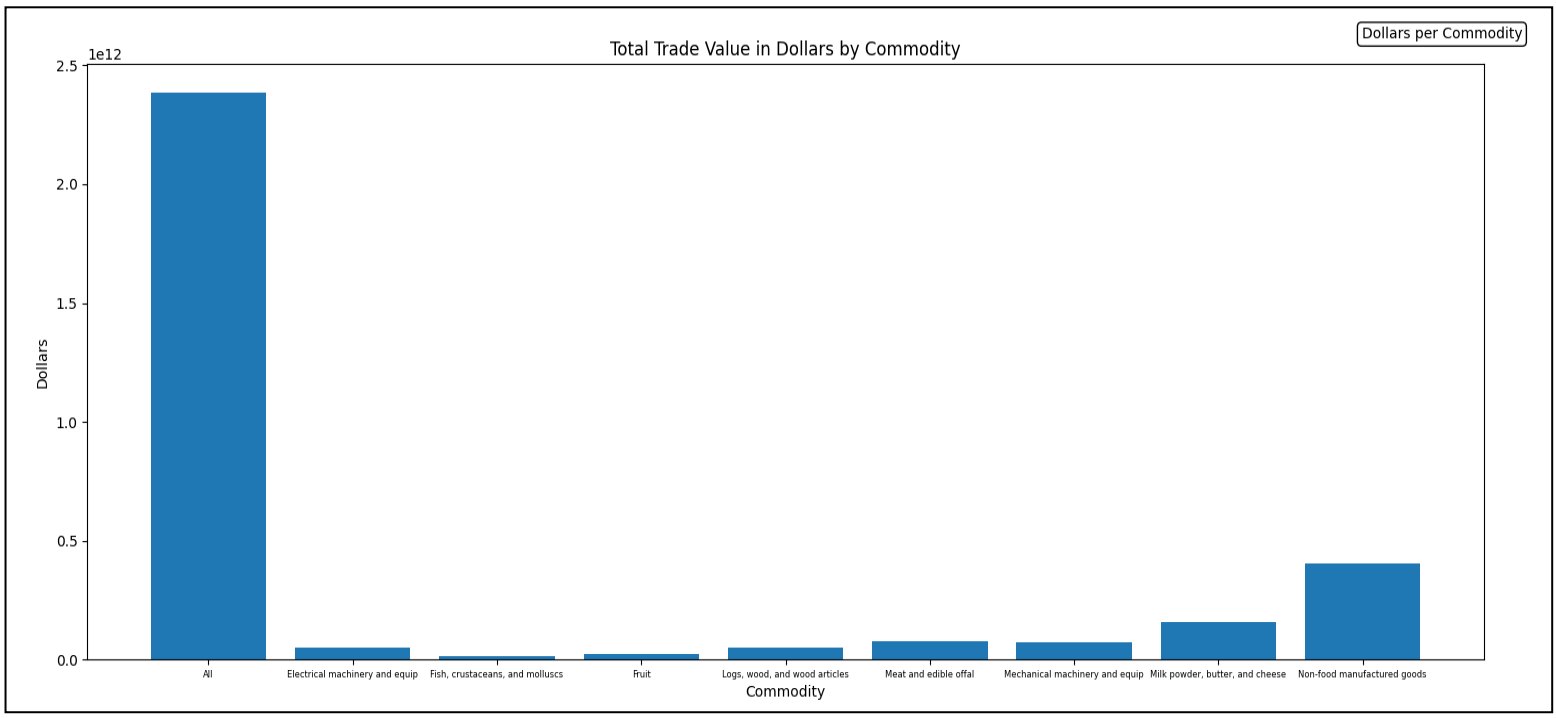
3.4. Συνολική παρουσίαση του τζίρου (στήλη value) για κάθε μέρα της εβδομάδας (στις αντίστοιχες μονάδες μέτρησης)

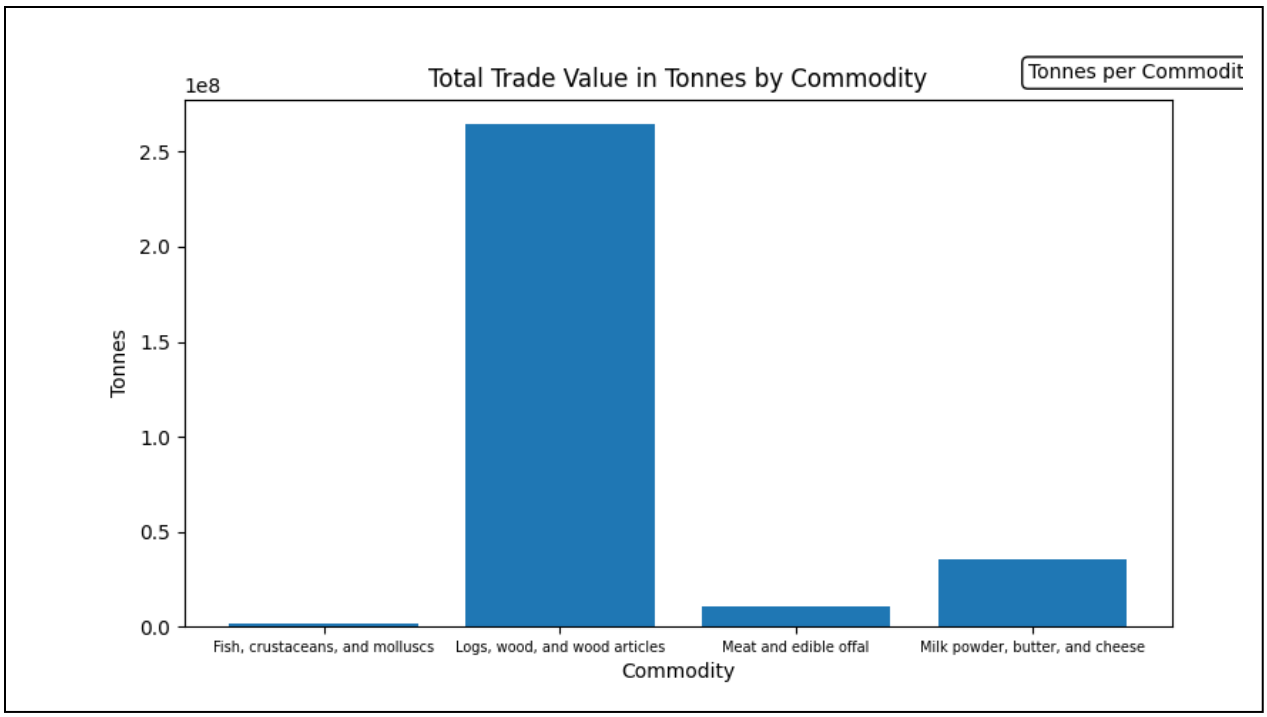
Ομοίως με το ερώτημα 3.1 κατασκευάσαμε τα παρακάτω γραφήματα:



3.5. Συνολική παρουσίαση του τζίρου (στήλη value) για κάθε κατηγορία εμπορεύματος (στις αντίστοιχες μονάδες μέτρησης)

Ομοίως με το ερώτημα 3.1 κατασκευάσαμε τα παρακάτω γραφήματα:



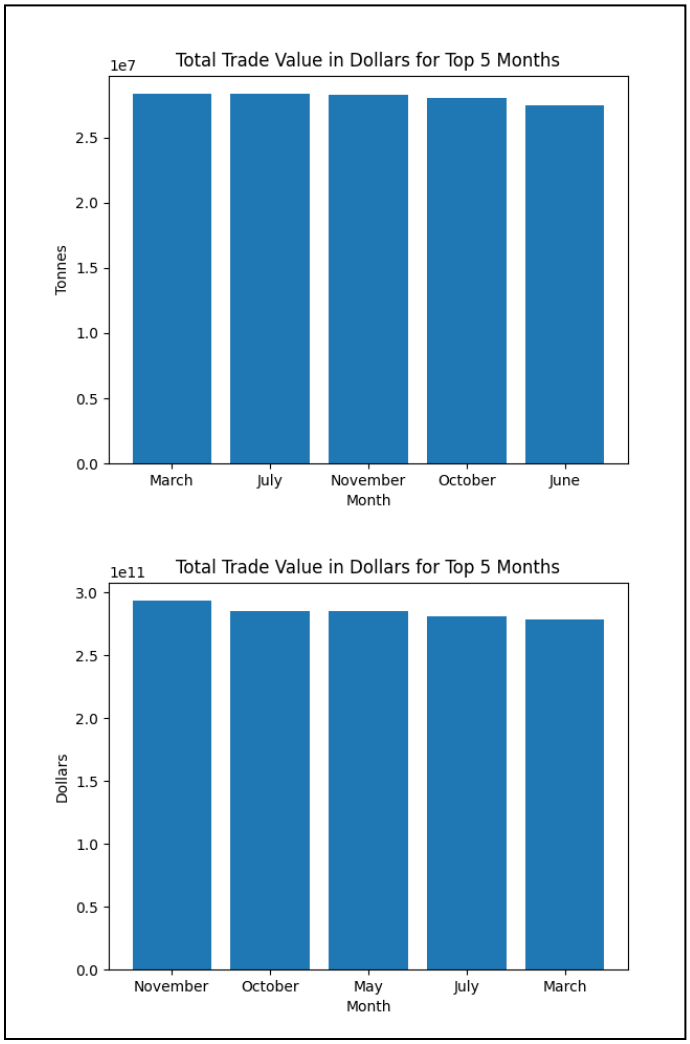


3.6. Παρουσίαση των 5 μηνών με το μεγαλύτερο τζίρο, ανεξαρτήτως μέσου μεταφοράς και είδους ανακυκλώσιμων ειδών

Για αυτό το ερώτημα θα εμφανίσουμε τους 5 Μήνες με το μεγαλύτερο τζίρο για την στήλη **Values** εμφανίζοντας ξεχωριστά τους μήνες για το '**$**' και για το '**Tonnes'**. Όπως στο ερώτημα 3.1 θα χωρίσουμε το dataset μεταξύ '**$**' και '**Tonnes'** και θα αθροίσουμε όλους τους όρους για κάθε μήνα.

Για να επιλεχθούν οι 5 μήνες με το μεγαλύτερο τζίρο θα χρειαστεί να χρησιμοποιήσουμε την συνάρτηση **zip** για να συνδυαστεί η λίστα **dollar_months** και η λίστα **month_names** όπου με την συνάρτηση **sorted** και με όρισμα **reverse=True** ταξινομούμε την λίστα σε φθίνουσα σειρά. Επίσης χρησιμοποιείται το [:5] για να εξαχθούν μόνο οι 5 πρώτοι μήνες με τα υψηλότερα ποσά δολαρίων. Αντίστοιχα και για την τιμή "**Tonnes**".

```
# Print the top 5 months by dollar amount
    month_names = list(calendar.month_name)[1:]
    dollar_top5 = sorted(zip(dollar_months, month_names), reverse=True)[:5]
    print("\n----------Top 5 Months by Dollars----------\n")
    for dollar_month, month_name in dollar_top5:
        print(f'{month_name} = {dollar_month}\n')
```



3.7. Παρουσίαση των 5 κατηγοριών εμπορευμάτων με το μεγαλύτερο τζίρο, για κάθε χώρα

Για να 5 κατηγοριών εμπορευμάτων με το μεγαλύτερο τζίρο, για κάθε χώρα θα χρειαστεί να υπολογίσουμε το άθροισμα ξεχωριστά για κάθε χώρα και εμπόρευμα των '**$**' και '**Tonnes'**. Αυτό το πετυχαίνουμε με την εντολή **df_dollar = df_dollar.groupby(['Country', 'Commodity'])['Value'].sum().reset_index()**.

Έπειτα, θα χρειαστεί να ταξινομήσουμε τα εμπορεύματα για κάθε χώρα σε φθίνουσα σειρά, έτσι ώστε να επιλέξουμε τα 5 πρώτα από αυτά. Αυτό γίνεται μέσω την εντολής **df_dollar = df_dollar.sort_values(['Country', 'Value'], ascending=[True, False])**.

Τέλος, θα χρειαστεί να εκτυπώσουμε αυτά τα αποτελέσματα, διατρέχοντας την λίστα και κάνοντας **print** την χώρα και τα **Commodities** μαζί με το **Value** τους.

```
# Group by country, commodity, and measure and sum 'Value'
    df_country_commodity = df.groupby(['Country', 'Commodity', 'Measure'])['Value'].sum().reset_index()


    # Filter for '$' measure and calculate total for each country and commodity
    df_dollar = df_country_commodity[df_country_commodity['Measure'] == '$'].groupby(['Country',
'Commodity'])['Value'].sum().reset_index()
    df_dollar = df_dollar.sort_values(['Country', 'Value'], ascending=[True, False])
    df_dollar_top5 = df_dollar.groupby('Country').head(5)


    print("\n----------Top 5 Commodities by Dollars----------\n")
    # Print the top 5 commodities for each country based on their value in dollars
    for country in df_dollar_top5['Country'].unique():
        top5 = df_dollar_top5[df_dollar_top5['Country'] == country]
        print(f"{country}:")
        for i, row in top5.iterrows():
            print(f"{row['Commodity']} = {row['Value']}")
        print()
```

Ο παραπάνω κώδικας έχει υλοποιηθεί για το Measure ‘$’ αλλά με παρόμοιο τρόπο είναι δομημένος και ο κώδικας για το ‘Tonnes’.

Παρακάτω φαίνεται τι θα εμφανιστεί στο περιβάλλον υλοποίησης του κώδικα όταν πατηθεί το κουμπί ‘Top 5 Commodities by country’.

----------Top 5 Commodities by Dollars----------

All:

All = 1603472000000

Non-food manufactured goods = 403154000000

Milk powder, butter, and cheese = 98757000000

Mechanical machinery and equip = 57567000000

Meat and edible offal = 51206000000

Australia:

All = 107686000000

China:

All = 182406000000

Milk powder, butter, and cheese = 31216000000

Logs, wood, and wood articles = 17993000000

Electrical machinery and equip = 16478000000

Meat and edible offal = 15463000000

East Asia (excluding China):

All = 89245000000

Milk powder, butter, and cheese = 27311000000

European Union (27):

All = 26644000000

Japan:

All = 23155000000

Total (excluding China):

All = 291991000000

United Kingdom:

All = 21591000000

United States:

All = 40477000000

Meat and edible offal = 11843000000

----------Top 5 Commodities by Tonnes----------

All:

Logs, wood, and wood articles = 154650000

Milk powder, butter, and cheese = 22118000

Meat and edible offal = 6749000

Fish, crustaceans, and molluscs = 1832000

China:

Logs, wood, and wood articles = 109752000

Milk powder, butter, and cheese = 7536000

Meat and edible offal = 2285000

East Asia (excluding China):

Milk powder, butter, and cheese = 6137000

United States:

Meat and edible offal = 1338000

3.8. Παρουσίαση της ημέρας με το μεγαλύτερο τζίρο, για κάθε κατηγορία εμπορεύματος

Αρχικά, θα χρειαστεί όπως και στα προηγούμενα ερωτήματα να βρούμε το άθροισμα του τζίρου για κάθε εμπόρευμα και κάθε μέρα. Θα χρειαστεί για κάθε Measure να υπολογιστεί ξεχωριστά για κάθε εμπόρευμα το άθροισμα του τζίρου και για κάθε μέρα.

Έπειτα, από τις διαθέσιμες μέρες που υπολογίστηκαν πρέπει να διαλέξουμε την ημέρα με τον μεγαλύτερο τζίρο για αυτό το εμπόρευμά.

```
#Group by weekday, commodity, and measure and sum 'Value'
    df_weekday = df.groupby(['Weekday', 'Commodity', 'Measure'])['Value'].sum().reset_index()


    #Filter for '$' measure and find the weekday with the max value for each commodity
    print("\n----------Max Dollar Weekdays by Commodity----------\n")
    for commodity in df_weekday[df_weekday['Measure'] == '$']['Commodity'].unique():
        max_dollar_row = df_weekday[(df_weekday['Commodity'] == commodity) &
(df_weekday['Measure'] == '$')].nlargest(1, 'Value')
        max_dollar_weekday = max_dollar_row['Weekday'].iloc[0]
        print(f'{commodity}: {max_dollar_weekday}')
```

Παρακάτω φαίνεται τι θα εμφανιστεί στο περιβάλλον υλοποίησης του κώδικα όταν πατηθεί το κουμπί ‘Top Commodity by day’.

```
----------Max Dollar Weekdays by Commodity----------

All: Monday
Electrical machinery and equip: Thursday
Fish, crustaceans, and molluscs: Thursday
Fruit: Friday
Logs, wood, and wood articles: Friday
Meat and edible offal: Sunday
Mechanical machinery and equip: Thursday
Milk powder, butter, and cheese: Monday
Non-food manufactured goods: Thursday
```

```
----------Max Tonne Weekdays by Commodity----------

Fish, crustaceans, and molluscs: Monday
Logs, wood, and wood articles: Friday
Meat and edible offal: Sunday
Milk powder, butter, and cheese: Monday
```

4. Σύνδεση με Βάση Δεδομένων.

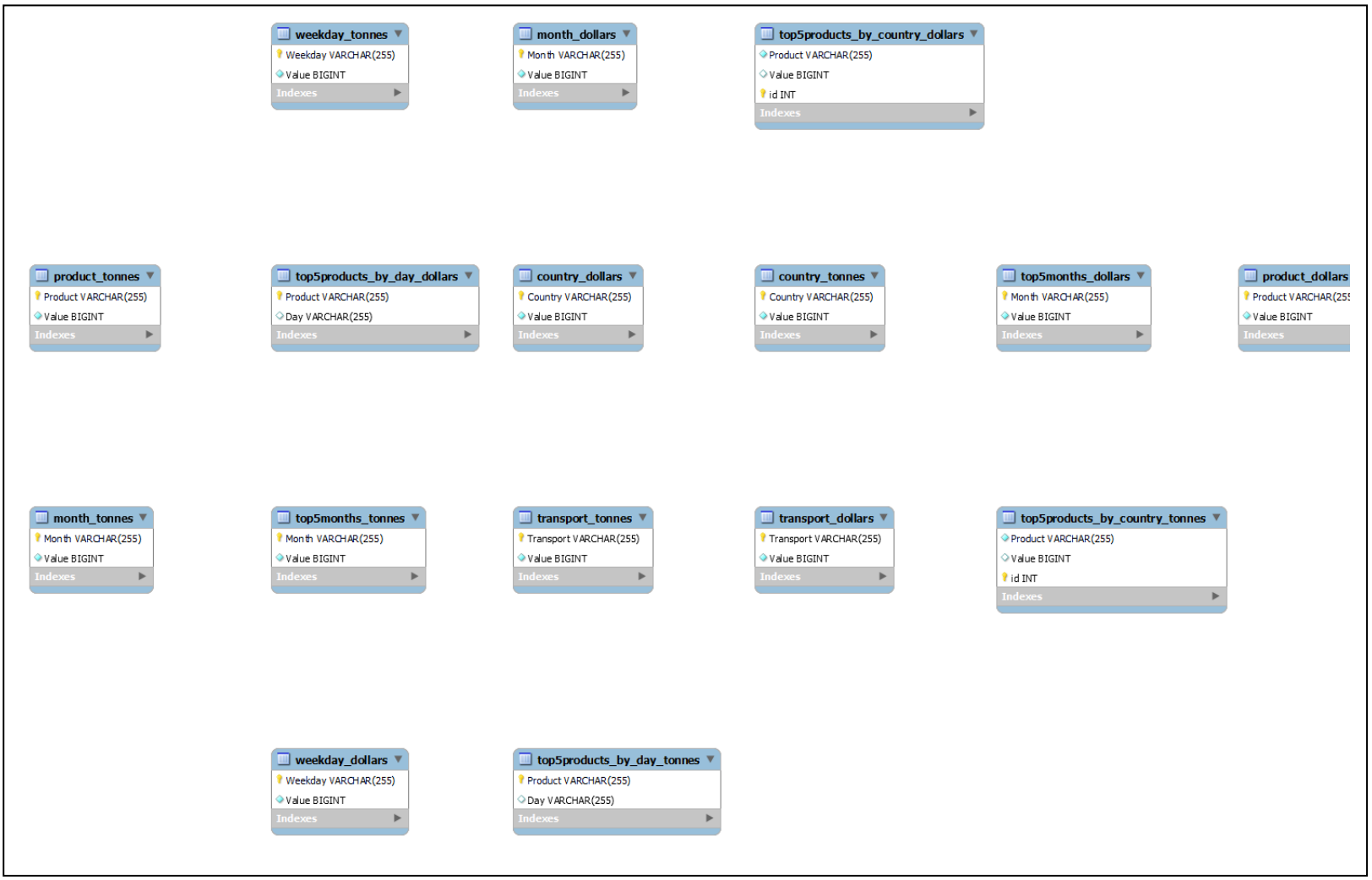
Για την σύνδεση των αποτελεσμάτων με μία **SQL** βάση θα χρειαστεί να προσθέσουμε την βιβλιοθήκη **mysql.connector** διότι θα χρησιμοποιήσουμε **MySQL**. Για να έχουμε πρόσβαση στην βάση θα χρειαστεί να αρχικοποιήσουμε έναν **cursor** με τον οποίο θα κάνουμε execute στην **MySQL**.

```
mydb = mysql.connector.connect(
  host="localhost",
  user="root",
  passwd="python"
)

mycursor = mydb.cursor()

mycursor.execute("CREATE DATABASE IF NOT EXISTS mydatabase;")
mycursor.execute("USE mydatabase;")
```

Έπειτα, θα χρειαστεί να δημιουργήσουμε τους πίνακες που θα αποθηκεύσουμε τα δεδομένα αλλά επίσης θα χρειαστεί μέσω της **INSERT** να προσθέσουμε δεδομένα στους πίνακες. Είναι επίσης σημαντικό πριν σταματήσουμε την σύνδεση με την **MySQL** να κάνουμε **commit** για να πραγματοποιηθούν οι αλλαγές στην **MySQL**.

```
mycursor.execute("CREATE TABLE IF NOT EXISTS month_dollars (Month VARCHAR(255) NOT NULL,Value bigint(10) NOT NULL,PRIMARY KEY(Month));")
mycursor.execute("CREATE TABLE IF NOT EXISTS month_dollars (Month VARCHAR(255) NOT NULL,Value bigint(10) NOT NULL,PRIMARY KEY(Month));")
…
…
…
mydb.commit()
```

Σχήμα βάσης:



5. Παραδοχές

- Για το 6ο ερώτημα που αφορά τους 5 μήνες δεν πάρθηκαν υπόψιν τα μέσα μεταφοράς και τα ανακυκλώσιμα είδη.
- Για το ερώτημα 6 δεν έγινε γράφος διότι θα χρειαζόταν 3d γράφος αφού θέλουμε για κάθε χώρα ξεχωριστά να δείξουμε τα top 5 προϊόντα, άρα κάποια προϊόντα θα εμφανίζονταν 2 φορές στον x-axis
- Για το ερώτημα 7 δεν έγινε γράφος διότι απλά εμφανίζονται οι μέρες της εμβδομάδας.