Τμήμα Μηχανικών Η/Υ & Πληροφορικής - Πανεπιστήμιο Πατρών
Πολυτεχνική Σχολή

Μάθημα: Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών
Τομέας Λογικού των Υπολογιστών

Διδάσκοντες: Ιωάννης Γαροφαλάκης, Σπυρίδων Σιούτας, Παναγιώτης Χατζηδούκας , Γεράσιμος Βονιτσάνος
Ακαδημαϊκό Έτος: 2022 – 2023
Ημ/νία παράδοσης: 18/06/2023

Εργαστηριακή Άσκηση στην Python

Εαρινό Εξάμηνο 2023

Ονοματεπώνυμο: Μηλτιάδης Μαντές
Α.Μ.: 1084661
E – mail: up1084661@upnet.gr
Εξάμηνο: ΣΤ'

Περιεχόμενα

1. Εισαγωγή

1.1 Σχόλια – Γενική Περιγραφή

Στόχος είναι να διαβάσουμε και να επεξεργαστούμε κατάλληλα τα δεδομένα του CSV αρχείου σύμφωνα με τα ερωτήματα που ζητούνται, φτιάχνοντας τις αντίστοιχες γραφικές παραστάσεις για καθένα από αυτά. Το αρχείο περιλαμβάνει εγγραφές που ακολουθούν την πιο κάτω δομή:

Direction | *Year* | *Date* | *Weekday* | *Country* | *Commodity* | *Transport_Mode* | *Measure* | *Value* | *Cumulative*

Ως **περιβάλλον υλοποίησης** χρησιμοποιούμε το περιβάλλον του PyCharm, το οποίο και κατεβάζουμε από το εργαλείο JetBrains.

Τέλος, επιλέγουμε η **υλοποίηση και η επεξεργασία της Βάσης Δεδομένων** όπου θα αποθηκεύουμε τα αποτελέσματα της επεξεργασίας των δεδομένων να πραγματοποιηθεί σε περιβάλλον Workbench της MySQL. Στο περιβάλλον αυτό θα μπορούμε να βλέπουμε τους ειδικά διαμορφωμένους πίνακες, από όπου και θα εξάγουμε τα τελικά δεδομένα σε καινούργια αρχεία CSV, ένα για κάθε ερώτημα που μας τίθεται.

Η **επικοινωνία του χρήστη** με τη Βάση Δεδομένων και τα γραφήματα εξασφαλίζεται με μια Γραφική Διεπαφή Χρήστη (GUI), όπου θα παρουσιάζεται ένα κατάλληλο μενού επιλογών και ο χρήστης θα επιλέγει ο ίδιος ποια ενέργεια επιθυμεί να πραγματοποιήσει.

1.2 Παραδοχές

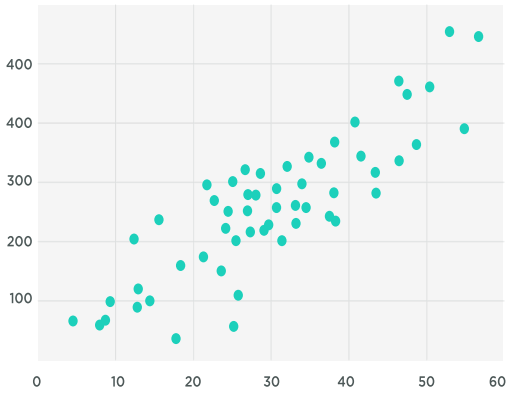
Ακολουθεί μια σύντομη περιγραφή του σκεπτικού πάνω στο οποίο στηρίχθηκε η υλοποίηση της συγκεκριμένης Εργαστηριακής Άσκησης:

1) **Πρώτο βήμα** αποτέλεσε η δημιουργία του παραθύρου root της Γραφικής Διεπαφής Χρήστη, το οποίο τροποποιήθηκε με κατάλληλες ρυθμίσεις, καθώς και η δημιουργία ενός σετ ρυθμίσεων το οποίο εφαρμόστηκε πάνω σε όλα τα κουμπιά της Διεπαφής.
2) **Δεύτερο βήμα** αποτέλεσε η σύνδεση του προγράμματος με τη Βάση Δεδομένων και η συγκέντρωση των δεδομένων από το αρχείο CSV από το αντίστοιχο url που μας μεταφέρει στην ιστοσελίδα που χρειαζόμαστε.
3) **Τρίτο βήμα** αποτέλεσε η δημιουργία 8 ξεχωριστών μεθόδων, μία για κάθε ερώτημα που τίθεται στην άσκηση, οι οποίες επεξεργάζονται τα δεδομένα, σχεδιάζουν τις αντίστοιχες γραφικές παραστάσεις και έπειτα ενημερώνουν τη Βάση Δεδομένων εισάγοντας στο σωστό πίνακα τα τελικά αποτελέσματα που προκύπτουν.
4) **Τέταρτο βήμα** αποτέλεσε ο ορισμός μιας ακόμα μεθόδου που εξάγει τα τελικά αποτελέσματα αφού περαστούν στη βάση Δεδομένων σε αντίστοιχα αρχεία CSV.
5) **Πέμπτο βήμα** και τελευταίο αποτέλεσε ο ορισμός μιας μεθόδου που εμφανίζει το μενού με τις διαθέσιμες επιλογές όταν ο χρήστης κάνει κλικ στο αντίστοιχο κουμπί του root το οποίο δημιουργείται στη συνέχεια.

Οι **βιβλιοθήκες** που έχουμε επιλέξει να χρησιμοποιήσουμε είναι οι: pandas, matplotlib.pyplot, tkinter, PIL και mysql.connector.

- **pandas:** φόρτωση και αποθήκευση των δεδομένων
- **matplotlib.pyplot**: σχεδίαση των γραφικών παραστάσεων
- **tkinter:** σχεδιασμός Γραφικής Διεπαφής Χρήστη
- **PIL**: φόρτωση και επεξεργασία εξωτερικού αρχείου εικόνας PNG
- **mysql.connector**: σύνδεση με τη Βάση Δεδομένων

Οι **γραφικές παραστάσεις** στα τελευταία τρία ερωτήματα έχουμε επιλέξει να ακολουθούν την μορφή ενός διαγράμματος διασποράς (scatter plot), καθώς είναι πολύ χρήσιμα όσον αφορά τη συσχέτιση δύο συνόλων, καθώς και τη μελέτη της επίδρασης των αλλαγών του ενός πάνω στο άλλο. Σε αυτές επίσης παρατίθεται και σχετικό υπόμνημα, γιατί οι πληροφορίες αφορούν παραπάνω από ένα αντικείμενα.



Οι **γραφικές παραστάσεις** στα πρώτα πέντε ερωτήματα έχουμε επιλέξει να είναι πιο απλές και να ακολουθούν την μορφή ραβδογράμματος (bar graph).

Για τη **Γραφική Διεπαφή Χρήστη** έχουμε επιλέξει να εμφανίζεται ένα αρχικό παράθυρο με ένα εισαγωγικό μήνυμα και δύο κουμπιά:

- **Show Graphs**: αν πατηθεί από τον χρήστη τον μεταφέρει σε καινούργιο παράθυρο από όπου μπορεί να πλοηγείται και επιλέξει την επόμενη ενέργεια που θέλει να ακολουθήσει ή να γυρίσει πίσω.
- **Exit**: αν πατηθεί από τον χρήστη η Διεπαφή κλείνει και ο χρήστης επιστρέφει πίσω στο περιβάλλον του PyCharm.

Το καινούργιο παράθυρο που αναφέρθηκε έχουμε επιλέξει επίσης να διαθέτει 10 κουμπιά, από τα οποία τα πρώτα οκτώ υλοποιούν τις λειτουργίες που αναφέρθηκαν και πιο πάνω. Ακόμα, έχουμε τα κουμπιά:

- **Export Data into CSV Files**: αν πατηθεί από τον χρήστη εξάγει όλα τα αρχεία CSV στο περιβάλλον του PyCharm.
- **Close**: αν πατηθεί από τον χρήστη, τότε επιστρέφει πίσω στο αρχικό μας παράθυρο.

Τέλος, για τη Βάση Δεδομένων έχουμε αποφασίσει να δημιουργήσουμε οκτώ πίνακες, ο καθένας από τους οποίους έχει την ακόλουθη δομή:

profit_per_month	{Profit BIGINT, Month INT, Measure VARCHAR(255)}
profit_per_country	{Profit BIGINT, Country VARCHAR(255), Measure VARCHAR(255)}
profit_per_transport_mean	{Profit BIGINT, Transport_Mean VARCHAR(255), Measure VARCHAR(255)}
profit_per_weekday	{Profit BIGINT, Weekday VARCHAR(255), Measure VARCHAR(255)}
profit_per_commodity	{Profit BIGINT, Country VARCHAR(255), Measure VARCHAR(255)}
top_5_months	{Month INT, Profit BIGINT, Measure VARCHAR(255)}
top_5_commodities	{Country VARCHAR(255), Commodity VARCHAR(255), Profit BIGINT, Measure VARCHAR(255)}
days_with_biggest_profit	{Commodity VARCHAR(255), Weekday VARCHAR(255), Profit BIGINT, Measure VARCHAR(255)}

Παρατηρήσεις:

1. Παρ' όλο που δεν αναφέρεται ρητά στην εκφώνηση ότι για τα ερωτήματα **6, 7**, **8** πρέπει να γίνει διαχωρισμός των μονάδων μέτρησης, έχει θεωρηθεί ότι πάλι εξάγονται δύο διαφορετικές γραφικές παραστάσεις, μία για $ και μια για *Tonnes*, όπως και στα προηγούμενα ερωτήματα.
2. Στο Κεφάλαιο 2 που ακολουθεί, οι μεταβλητές και οι μέθοδοι θα σημειώνονται με **έντονη γραφή**, ενώ οποιαδήποτε άλλα στοιχεία του κώδικα που δεν ανήκουν σε αυτές τις κατηγορίες με απαλή μπλε.

2. Ανάλυση Κώδικα σε Python

2.1 Γραφικές παραστάσεις

Σε αυτή την υποενότητα αναλύεται η υλοποίηση των οκτώ μεθόδων που χρησιμοποιούμε για να εκτελέσουμε τα ερωτήματα της άσκησης. Η κάθε μέθοδος έχει όνομα:

graph_i( ) , i = 1,2,3,4,5,6,7,8

και αντιπροσωπεύει το ερώτημα στο οποίο αναφερόμαστε κάθε φορά.
Ακολουθεί ενδεικτικός σχολιασμός για την υλοποίηση της μεθόδου **graph_1( )**, ωστόσο και για τις υπόλοιπες με **i = 2,3,4,5** η διαδικασία που εφαρμόζεται ακολουθεί το ίδιο σκεπτικό.

```
def graph_1():
    # Μετατρέπουμε το αλφαριθμητικό που αντιπροσωπεύει την ημερομηνία στη μορφή datetime
    # με format=mixed γιατί δεν είναι της μορφής day-month-year
    data['Date'] = pd.to_datetime(data['Date'], format='mixed')

    # Ομαδοποιούμε τα δεδομένα βάσει μήνα και υπολογίζουμε το τζίρο για κάθε μήνα και measure
    grouped = data.groupby(['Measure', data['Date'].dt.month])['Value'].sum().reset_index()

    # Βρίσκουμε τις διακριτές τιμές του measure
    unique_measures = grouped['Measure'].unique()

    # Διαβάζουμε τα δεδομένα για κάθε διαφορετική τιμή του measure
    for measure in unique_measures:
        # Ομαδοποιούμε τα δεδομένα για κάθε διαφορετική τιμή του measure
        measure_data = grouped[grouped['Measure'] == measure]

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        plt.bar(measure_data['Date'], measure_data['Value'])
        plt.title(f"Profit per Month ({measure})")
        plt.xlabel("Months")
        plt.ylabel("Profits")
        plt.xticks(measure_data['Date'])
        plt.tight_layout()
        plt.show()
```

Πάνω στον κώδικα έχουν προστεθεί και κατάλληλα σχόλια για να καθιστούν την κατανόησή του πιο εύκολη.

Αρχικά, στο **data['Date']** αποθηκεύουμε όλα τα αλφαριθμητικά που αντιπροσωπεύουν τις ημερομηνίες, ωστόσο επειδή θέλουμε να απομονώσουμε τους μήνες τα φέρνουμε σε μορφή datetime μέσω της μεθόδου **to_datetime( )**. Έπειτα, στην μεταβλητή **grouped** αποθηκεύουμε τα ομαδοποιημένα δεδομένα μέσω της πιο κάτω διαδικασίας:

1. **data['Date'].dt.month** : απομονώνει το κομμάτι του μήνα από όλη την ημερομηνία εφόσον είναι στη μορφή datetime στη στήλη Date και το γνώρισμα dt.month επιστρέφει έναν ακέραιο που αντιπροσωπεύει το μήνα.

2. **data.groupby(['Measure', data['Date'].dt.month])**: ομαδοποιεί τα δεδομένα με βάσει τους μοναδικούς συνδυασμούς της στήλης Measure και τις τιμές των μηνών που ορίσαμε πιο πάνω.
3. **['Value'].sum().reset_index()** : υπολογίζει το άθροισμα των τιμών της στήλης Value για κάθε ομάδα που προέκυψε από το βήμα 2 και μέσω της **reset_index()** μετατρέπονται τα δεδομένα πάλι σε DataFrame.

Στη συνέχεια, εφαρμόζοντας τη μέθοδο **unique( )** πάνω στη **grouped** λαμβάνουμε όλες τις διακριτές τιμές που λαμβάνει η στήλη Measure (*$, Tonnes*), εφόσον θέλουμε. διαφορετική γραφική για κάθε μονάδα μέτρησης.

Έπειτα, για κάθε τιμή του μετρητή **measure** με έναν βρόγχο επανάληψης, εφόσον στη **grouped** υπάρχουν τα επιθυμητά δεδομένα, τα ομαδοποιούμε πάλι με βάση τη τιμή του **measure** και τα εκχωρούμε στη **measure_data**.

Τέλος, για να σχεδιάσουμε τη γραφική παράσταση, χρησιμοποιούμε μέσα στον ίδιο βρόγχο επανάληψης τη μέθοδο **bar( )** για τη χάραξη και άλλες όπως η **title( )**, **xlabel()**, **ylabel( )** κ.τ.λ. για την μορφοποίηση του γραφήματος. Μέσω της **show( )** εμφανίζεται το τελικά διαμορφωμένο γράφημα σε ένα καινούργιο παράθυρο.

Σημαντικό είναι να τονίσουμε σε αυτό το σημείο πως στον πίνακα data από όπου επεξεργαζόμαστε τα δεδομένα μας έχουμε ήδη εισάγει τα στοιχεία του CSV αρχείου μέσω του εξής κώδικα:

```
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'
data = pd.read_csv(url)
```

Στις υπόλοιπες τέσσερις μεθόδους ακολουθούμε ακριβώς την ίδια διαδικασία κάνοντας μικρές αλλαγές όπου χρειαστεί, όπως στα ορίσματα της **groupby( ),** ανάλογα με το ποιο στοιχείο θέλουμε να ομαδοποιήσουμε κάθε φορά. Η χάραξη της γραφικής δεν υφίσταται καμία τροποποίηση σε καμία από αυτές παρά μόνον στην επιλογή της στήλης από τη **measure_data** που σχεδιάζουμε. Ενδεικτικά παραθέτουμε και την **graph_2( )**:

```
def graph_2():
    grouped = data.groupby(['Country', 'Measure'])['Value'].sum().reset_index()
    unique_measures = grouped['Measure'].unique()

    for measure in unique_measures:
        measure_data = grouped[grouped['Measure'] == measure]

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        plt.bar(measure_data['Country'], measure_data['Value'])
        plt.title(f"Profit per Country ({measure})")
        plt.xlabel("Countries")
        plt.ylabel("Profits")
        plt.xticks(measure_data['Country'], rotation=45)
```

```
    plt.tight_layout()
    plt.show()
```

Αφού τελειώσαμε με τις πέντε πρώτες μεθόδους, μεταβαίνουμε τώρα στον σχολιασμό της μεθόδου **graph_6( )**, η οποία αποθηκεύει και σχεδιάζει τους πέντε μήνες με το μεγαλύτερο τζίρο. Το βασικό σκεπτικό της υλοποίησης δεν διαφέρει και πολύ από τις προηγούμενες πέντε μεθόδους, ωστόσο είναι απαραίτητο να γίνουν οι παρακάτω δύο προσθήκες, τις οποίες θα αναλύσουμε ευθύς αμέσως:

```
# Βρίσκουμε τους 5 μήνες με το μεγαλύτερο τζίρο
top_5_months = (
measure_data.groupby('Measure')
.apply(lambda x: x.nlargest(5, 'Value'))
.reset_index(drop=True)
)

plt.figure(figsize=(10, 6))
for _, data in top_5_months.iterrows():
  plt.bar(data['Date'], data['Value'])
```

Στο πρώτο κομμάτι κώδικα, το οποίο τοποθετούμε επίσης μέσα στον βρόγχο επανάληψης, θέλουμε από τα ομαδοποιημένα με βάση τη τιμή του **measure** δεδομένα στη **measure_data** να κρατήσουμε τις χώρες με τις πέντε υψηλότερες τιμές της στήλης Value. Αυτό πραγματοποιείται μέσω της εφαρμογής της **groupby('Measure')** στη **measure_data** και ταυτόχρονα της **apply(lambda x: x.nlargest(5, 'Value'))** πάνω σε αυτή, η οποία εξάγει τις πέντε κορυφαίες τιμές. Στο δεύτερο κομμάτι κώδικα (και αυτό μέσα στο βρόγχο επανάληψης) με έναν ακόμα βρόχο επανάληψης τοποθετούμε κάθε έναν από τους πέντε αυτούς μήνες στο γράφημα μέσω της **bar( ).**

Άρα, η τελική μορφή της **graph_6( )** θα προκύψει ως ακολούθως :

```
def graph_6():
    data = pd.read_csv(url)
    data['Date'] = pd.to_datetime(data['Date'], format='mixed')
    grouped = data.groupby(['Measure', data['Date'].dt.month])['Value'].sum().reset_index()

    unique_measures = grouped['Measure'].unique()

    for measure in unique_measures:
        measure_data = grouped[grouped['Measure'] == measure]

        # Βρίσκουμε τους 5 μήνες με το μεγαλύτερο τζίρο
        top_5_months = (
        measure_data.groupby('Measure')
        .apply(lambda x: x.reset_index(drop=True).nlargest(5, 'Value'))
        .reset_index(drop=True)
        )

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        for _, data in top_5_months.iterrows():
            plt.bar(data['Date'], data['Value'])

        plt.title(f'Top 5 Months with the Biggest Profit ({measure})')
```

```
        plt.xlabel("Months")
        plt.ylabel("Profits")
        plt.xticks(measure_data['Date'])
        plt.tight_layout()
    plt.show()
```

Η **graph_7( )** αποθηκεύει και σχεδιάζει τα κορυφαία πέντε εμπορεύματα για κάθε χώρα με βάση τον τζίρο τους. Υλοποιείται ακριβώς όμοια προσθέτοντας στον παραπάνω κώδικα τα κομμάτια κώδικα:

```
# Βρίσκουμε τις 5 κατηγορίες προϊόντων με το μεγαλύτερο τζίρο ανά χώρα
top_5_commodities = (
measure_data.groupby(['Measure', 'Country'])
apply(lambda x: x.nlargest(5, 'Value'))
reset_index(drop=True)
)

for country, data in top_5_commodities.groupby('Country'):
        plt.scatter(data['Commodity'], data['Value'], marker='o', label=f"{country}")
```

Στο πρώτο κομμάτι κώδικα, το οποίο τοποθετούμε επίσης μέσα στον βρόγχο επανάληψης, θέλουμε τα ομαδοποιημένα με βάση τη τιμή του **measure** δεδομένα στη **measure_data** να ταξινομηθούν με βάση τη χώρα και να κρατήσουμε τις χώρες με τις πέντε υψηλότερες τιμές της στήλης Value. Αυτό πραγματοποιείται μέσω της εφαρμογής της **groupby(['Measure', 'Country'])** στη **measure_data** και ταυτόχρονα της **apply(lambda x: x.nlargest(5, 'Value'))** πάνω σε αυτή, η οποία εξάγει τις πέντε κορυφαίες τιμές. Στο δεύτερο κομμάτι κώδικα (και αυτό μέσα στο βρόγχο επανάληψης) με έναν ακόμα βρόχο επανάληψης για κάθε χώρα βρίσκουμε τις κορυφαίες πέντε κατηγορίες προϊόντων και τις χαράσσουμε σε γράφημα μέσω της **scatter( ).**

Άρα, η τελική μορφή της **graph_7( )** θα προκύψει ως ακολούθως:

```
def graph_7():
    data = pd.read_csv(url)
    grouped = data.groupby(['Measure', 'Country', 'Commodity'])['Value'].sum().reset_index()

    unique_measures = grouped['Measure'].unique()

    for measure in unique_measures:
        measure_data = grouped[grouped['Measure'] == measure]

        # Βρίσκουμε τις 5 κατηγορίες προϊόντων με το μεγαλύτερο τζίρο ανά χώρα
        top_5_commodities = (
            measure_data.groupby(['Measure', 'Country'])
            .apply(lambda x: x.nlargest(5, 'Value'))
            .reset_index(drop=True)
        )

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        for country, data in top_5_commodities.groupby('Country'):
            plt.scatter(data['Commodity'], data['Value'], marker='o', label=f"{country}")

        plt.title(f"Top 5 Commodities with the Biggest Profit per Country ({measure})")
```

```
    plt.xlabel("Commodities")
    plt.ylabel("Profits")
    plt.xticks(rotation=45)
    plt.legend()
    plt.tight_layout()
plt.show()
```

Όμοια ακριβώς υλοποιείται και η **graph_8( )** προσθέτοντας στον παραπάνω κώδικα τα κομμάτια κώδικα:

```
max_days = measure_data.groupby('Commodity')['Value'].idxmax()
max_days_data = measure_data.loc[max_days]

for commodity, data in max_days_data.groupby('Commodity'):
        plt.plot(data['Date'], data['Value'], marker='o', label=f'{commodity}')
```

Στο πρώτο κομμάτι βρίσκουμε τις μέρες με τον μεγαλύτερο τζίρο και αποθηκεύουμε τα δεδομένα τους στην **max_days_data**. Έπειτα, προσθέτουμε το δεύτερο κομμάτι εντός ενός ακόμα βρόγχου επανάληψης και για κάθε μέρα βρίσκουμε τα προϊόντα που της αντιστοιχούν και τα χαράσσουμε και αυτά πάνω στη γραφική μας παράσταση.

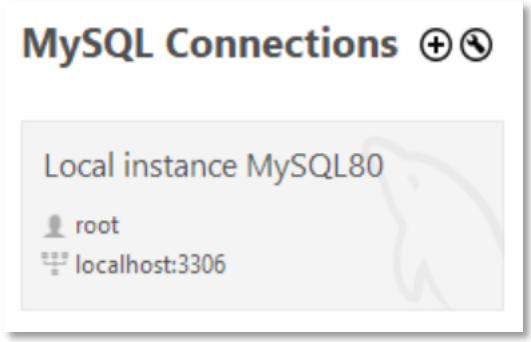
Παρατήρηση:
Μέσω των ορισμάτων **marker='o', label=f'{commodity}** στις **scatter( )** και **plot( )** δημιουργούμε και κατάλληλο υπόμνημα για την καλύτερη κατανόηση της γραφικής παράστασης.

Ενδεικτικά παρατίθεται και ο τελικός κώδικας της **graph_8( )**:

```
def graph_8():
    data = pd.read_csv(url)
    grouped = data.groupby(['Measure', 'Commodity', 'Date'])['Value'].sum().reset_index()

    unique_measures = grouped['Measure'].unique()

    for measure in unique_measures:
        measure_data = grouped[grouped['Measure'] == measure]

        # Βρίσκουμε τις μέρες με το μεγαλύτερο τζίρο για κάθε προιόν
        max_days = measure_data.groupby('Commodity')['Value'].idxmax()
        max_days_data = measure_data.loc[max_days]

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        for commodity, data in max_days_data.groupby('Commodity'):
            plt.plot(data['Date'], data['Value'], marker='o', label=f'{commodity}')

        plt.title(f"Days with the Biggest Profit per Commodity ({measure})")
        plt.xlabel("Days")
        plt.ylabel("Profits")
        plt.xticks(rotation=45)
        plt.legend()
        plt.tight_layout()
    plt.show()
```

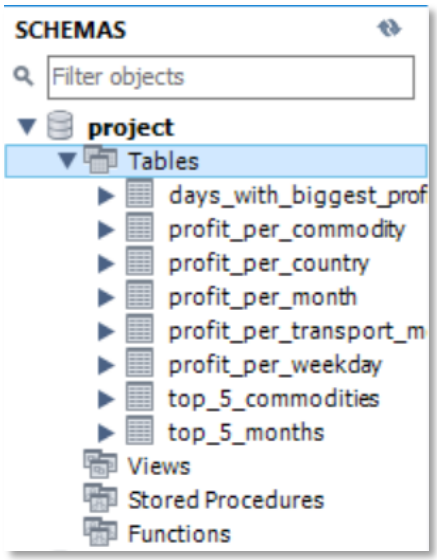
2.2 Σύνδεση με Βάση Δεδομένων σε MySQL

Σε αυτή την υποενότητα αναλύεται η δημιουργία και η σύνδεση της Β.Δ. με το πρόγραμμά μας. Πρώτο μας μέλημα αποτελεί να συνδέσουμε το πρόγραμμά μας με το Workbench της MySQL, στο οποίο έχουμε δημιουργήσει τη βάση με όνομα project. Αυτό υλοποιείται εύκολα αν μέσω της **connect( )** συνδέσουμε τη βάση μας με τα πιο κάτω στοιχεία στο αντικείμενο **mydb**. Για να επιτύχουμε και αλληλεπίδραση με τη Β.Δ. ορίζουμε ένα αντικείμενο με όνομα **cursor** τύπου **cursor( )** και το καλούμε πάνω στο αντικείμενο **mydb** που ορίσαμε πριν.




```
mydb = mysql.connector.connect(host="localhost", user="root", passwd="ju59GSX4yn",
database="project")
cursor = mydb.cursor()
```

Αφού επιτευχθεί η σύνδεση, πρέπει να δημιουργήσουμε τους πίνακες της βάσης, γι' αυτό και εκτελούμε την **execute( )** πάνω στο query που υλοποιεί την εντολή MySQL CREATE TABLE. Για παράδειγμα, για να δημιουργήσουμε τον πρώτο πίνακα θέτουμε ως query το εξής:




```
query = "CREATE TABLE profit_per_month (Profit BIGINT, Month INT,
Measure VARCHAR(255))"
cursor.execute(query)
```

Οι προδιαγραφές των πινάκων, δηλαδή πόσες στήλες θα έχουν, πώς αυτές θα ονομάζονται και τί είδους στοιχεία θα περιλαμβάνουν έχουν σημειωθεί στο **Κεφάλαιο 1.2, σελίδα 4**, οπότε δεν τις ξανά αναφέρουμε. Όμοια με τον πρώτο πίνακα δημιουργούνται και όλοι οι υπόλοιποι πίνακες της Β.Δ..

Αφού ορίσαμε τους πίνακες, σειρά έχει τώρα να εισάγουμε σε αυτούς τα αποτελέσματα που προέκυψαν από την επεξεργασία των δεδομένων, όπως αυτή περιεγράφηκε στο **Κεφάλαιο 2.1**. Για να γίνει αυτό, μεταβαίνουμε στο κώδικα της μεθόδου **graph_1( )** και διατρέχουμε τη **measure_data** για κάθε τιμή του μετρητή **measure** μέσω της μεθόδου **iterrows()**. Για κάθε μία από τις στήλες της **row** αποθηκεύουμε στις μεταβλητές **month**, **value** τις τιμές που έχουν αντίστοιχα οι στήλες Month, Value. Έτσι, ο κώδικας της **graph_1( )** στο τέλος θα δείχνει ως εξής:

```
def graph_1():
    # Μετατρέπουμε το αλφαριθμητικό που αντιπροσωπεύει την ημερομηνία στη μορφή datetime
    # με format=mixed γιατί δεν είναι της μορφής day-month-year
    data['Date'] = pd.to_datetime(data['Date'], format='mixed')

    # Ομαδοποιούμε τα δεδομένα βάσει μήνα και υπολογίζουμε το τζίρο για κάθε μήνα και measure
    grouped = data.groupby(['Measure', data['Date'].dt.month])['Value'].sum().reset_index()
```

```
    # Βρίσκουμε τις διακριτές τιμές του measure
    unique_measures = grouped['Measure'].unique()

    # Διαβάζουμε τα δεδομένα για κάθε διαφορετική τιμή του measure
    for measure in unique_measures:
        # Ομαδοποιούμε τα δεδομένα για κάθε διαφορετική τιμή του measure
        measure_data = grouped[grouped['Measure'] == measure]

        for index, row in measure_data.iterrows():
            month = row['Date']
            value = row['Value']
            cursor.execute("INSERT INTO profit_per_month (Profit, Month, Measure) VALUES (%s, %s, %s)",
                           (value, month, measure))

        # Εισάγουμε τις αλλαγές που πραγματοποιήσαμε στον cursor της ΒΔ και κλείνει η σύνδεση
        mydb.commit()

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        plt.bar(measure_data['Date'], measure_data['Value'])
        plt.title(f"Profit per Month ({measure})")
        plt.xlabel("Months")
        plt.ylabel("Profits")
        plt.xticks(measure_data['Date'])
        plt.tight_layout()
        plt.show()
```

Μέσω της **execute( )** πραγματοποιούμε την εντολή εκχώρησης της MySQL INSERT όπου εισάγονται στις στήλες Profit, Month, Measure οι τιμές των **value**, **month** και **measure**. Επιβεβαιώνουμε τις αλλαγές αυτές καλώντας την **commit( )** στο αντικείμενο **mydb** που αντιπροσωπεύει τη βάση μας. Ακολουθούμε ακριβώς την ίδια διαδικασία και για τις υπόλοιπες τέσσερις μεθόδους που ακολουθούν, μεταβάλλοντας μόνον τα στοιχεία που εισάγουμε στους εκάστοτε πίνακες κάθε φορά. Ενδεικτικά παραθέτουμε πως θα μεταβληθεί και η **graph_2( )** μετά τις αλλαγές:

```
def graph_2():
    grouped = data.groupby(['Country', 'Measure'])['Value'].sum().reset_index()
    unique_measures = grouped['Measure'].unique()

    for measure in unique_measures:
        measure_data = grouped[grouped['Measure'] == measure]

        for index, row in measure_data.iterrows():
            country = row['Country']
            value = row['Value']
            cursor.execute("INSERT INTO profit_per_country (Profit, Country, Measure) VALUES (%s, %s, %s)",
                           (value, country, measure))

        # Εισάγουμε τις αλλαγές που πραγματοποιήσαμε στον cursor της ΒΔ και κλείνει η σύνδεση
        mydb.commit()

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        plt.bar(measure_data['Country'], measure_data['Value'])
        plt.title(f"Profit per Country ({measure})")
```

```
    plt.xlabel("Countries")
    plt.ylabel("Profits")
    plt.xticks(measure_data['Country'], rotation=45)
    plt.tight_layout()
    plt.show()
```

Για την **graph_6( )** θέλουμε για κάθε τιμή της **measure** και για κάθε σειρά **row** της **top_5_months** να εισάγουμε στις στήλες Month, Profit, Measure του πίνακα της Β.Δ. τις τιμές της στήλης Date, Value, Measure της **top_5_months.** Αυτό πραγματοποιείται μέσω της **execute( )** πάνω στο query που δημιουργούμε για την εντολή εκχώρησης της MySQL INSERT και επιβεβαιώνουμε τις αλλαγές αυτές καλώντας την **commit( )** στο αντικείμενο **mydb**. Έτσι, ο κώδικας θα διαμορφωθεί ως εξής:

```
def graph_6():
    data = pd.read_csv(url)
    data['Date'] = pd.to_datetime(data['Date'], format='mixed')
    grouped = data.groupby(['Measure', data['Date'].dt.month])['Value'].sum().reset_index()

    unique_measures = grouped['Measure'].unique()

    for measure in unique_measures:
        measure_data = grouped[grouped['Measure'] == measure]

        # Βρίσκουμε τους 5 μήνες με το μεγαλύτερο τζίρο
        top_5_months = (
        measure_data.groupby('Measure')
        .apply(lambda x: x.reset_index(drop=True).nlargest(5, 'Value'))
        .reset_index(drop=True)
        )

        for _, row in top_5_months.iterrows():
            query = 'INSERT INTO top_5_months (Month, Profit, Measure) VALUES (%s, %s, %s)'
            cursor.execute(query, (row['Date'], row['Value'], measure))
        # Εισάγουμε τις αλλαγές που πραγματοποιήσαμε στον cursor της ΒΔ και κλείνει η σύνδεση
        mydb.commit()

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        for _, data in top_5_months.iterrows():
            plt.bar(data['Date'], data['Value'])

        plt.title(f"Top 5 Months with the Biggest Profit ({measure})")
        plt.xlabel("Months")
        plt.ylabel("Profits")
        plt.xticks(measure_data['Date'])
        plt.tight_layout()
    plt.show()
```

Την ίδια μεθοδολογία ακολουθούμε και για τις άλλες δύο μεθόδους **graph_7( )**, **graph_8( )**, γι' αυτό δεν θα παραθέσουμε όλο τον κώδικά τους παρά μόνον το κομμάτι που προσθέσαμε σε κάθε μέθοδο τροποποιημένο:

Για την **graph_7( )** προσθέτουμε στο ίδιο σημείο:

```
for _, row in top_5_commodities.iterrows():
        query = 'INSERT INTO top_5_commodities (Country, Commodity, Profit, Measure) VALUES (%s, %s, %s, %s)'
        cursor.execute(query, (row['Country'], row['Commodity'], row['Value'], measure))
# Εισάγουμε τις αλλαγές που πραγματοποιήσαμε στον cursor της ΒΔ και κλείνει η σύνδεση
 mydb.commit()
```

Για την **graph_8( )** προσθέτουμε στο ίδιο σημείο:

```
for _, row in max_days_data.iterrows():
    query = 'INSERT INTO days_with_biggest_profit (Commodity, Weekday, Profit, Measure) VALUES (%s, %s, %s, %s)'
    cursor.execute(query, (row['Commodity'], row['Date'], row['Value'], measure))
# Εισάγουμε τις αλλαγές που πραγματοποιήσαμε στον cursor της ΒΔ και κλείνει η σύνδεση
mydb.commit()
```

Αφού ολοκληρώσαμε και τις οκτώ μεθόδους από πριν, πηγαίνουμε τώρα να εξετάσουμε και τη μέθοδο **export_csv( )**, η οποία εξάγει από τη Β.Δ. μας τα τελικά CSV αρχεία με τα δεδομένα που περιλαμβάνει ο κάθε πίνακας.

```
def export_csv():
    query = "SELECT * FROM profit_per_month"
    cursor.execute(query)
    res = cursor.fetchall()

    for row in res:
        new_data = pd.read_sql_query(query, mydb)
        new_data.to_csv('profits_per_month.csv', index=FALSE)

    query = "SELECT * FROM profit_per_country"
    cursor.execute(query)
    res = cursor.fetchall()

    for row in res:
        new_data = pd.read_sql_query(query, mydb)
        new_data.to_csv('profits_per_country.csv', index=FALSE)

    query = "SELECT * FROM profit_per_transport_mean"
    cursor.execute(query)
    res = cursor.fetchall()

    for row in res:
        new_data = pd.read_sql_query(query, mydb)
        new_data.to_csv('profits_per_transport_mode.csv', index=FALSE)

    query = "SELECT * FROM profit_per_weekday"
    cursor.execute(query)
    res = cursor.fetchall()

    for row in res:
        new_data = pd.read_sql_query(query, mydb)
        new_data.to_csv('profits_per_weekday.csv', index=FALSE)

    query = "SELECT * FROM profit_per_commodity"
```

```
cursor.execute(query)
res = cursor.fetchall()

for row in res:
    new_data = pd.read_sql_query(query, mydb)
    new_data.to_csv('profits_per_commodity.csv', index=FALSE)

query = "SELECT * FROM top_5_months"
cursor.execute(query)
res = cursor.fetchall()

for row in res:
    new_data = pd.read_sql_query(query, mydb)
    new_data.to_csv('top_5_months.csv', index=FALSE)

query = "SELECT * FROM top_5_commodities"
cursor.execute(query)
res = cursor.fetchall()

for row in res:
    new_data = pd.read_sql_query(query, mydb)
    new_data.to_csv('top_5_commodities.csv', index=FALSE)

query = "SELECT * FROM days_with_biggest_profit"
cursor.execute(query)
res = cursor.fetchall()

for row in res:
    new_data = pd.read_sql_query(query, mydb)
    new_data.to_csv('days_with_biggest_profit.csv', index=FALSE)
```

Αρχικά, ορίζουμε το query που θα περιλαμβάνει την εντολή MySQL SELECT, η οποία μας επιστρέφει τα περιεχόμενα όλων των στηλών του πίνακα. Έπειτα, καλούμε πάνω σε αυτό το query τη μέθοδο **execute( )** στον cursor που επικοινωνεί με τη βάση και τα αποτελέσματα που επιστρέφονται τα αποθηκεύουμε σε μια τοπική μεταβλητή **res** μέσω της μεθόδου **fetchall( )**. Τέλος, με έναν βρόγχο επανάληψης για κάθε σειρά row της λίστας **res** διαβάζουμε τα δεδομένα από τη βάση μέσω της **read_sql_query( )**, η οποία εκτελεί το query πάνω στη βάση και αποθηκεύει ό τι επιστρέφεται στη μεταβλητή **new_data**. Αφού γίνει αυτό καλούμε πάνω στη **new_data** τη μέθοδο **to_csv( )** με ορίσματα το όνομα του αρχείου που επιθυμούμε να δώσουμε, η οποία εξάγει το τελικό αρχείο.

Αυτή η διαδικασία επαναλαμβάνεται για κάθε πίνακα της Β.Δ. φτιάχνοντας κάθε φορά το αντίστοιχο query που πραγματοποιεί SELECT στον εκάστοτε πίνακα. Έτσι, θα έχουμε στο τέλος και τα οκτώ αρχεία που μας ζητούνται.

2.3 Γραφική Διεπαφή Χρήστη

Τελευταίο κομμάτι που απομένει για την ολοκλήρωση της άσκησης είναι η δημιουργία και η μορφοποίηση της Γραφικής Διεπαφής Χρήστη, η οποία θα του επιτρέπει να διαλέγει ο ίδιος από ένα μενού επιλογών ποια/ποιες γραφικές παραστάσεις των δεδομένων θέλει να εμφανιστούν. Το γενικό μοντέλο που θα ακολουθήσουμε όσον αφορά το λειτουργικό κομμάτι της Διεπαφής έχει παρουσιαστεί πιο πάνω στο **Κεφάλαιο 1.2, σελίδα 4**, οπότε περνάμε στο σχεδιαστικό κομμάτι.

Για να δημιουργήσουμε το αρχικό παράθυρο το οποίο θα βλέπει ο χρήστης με το που ανοίγει τη Διεπαφή, ξεκινάμε ορίζοντας τη ρίζα **root** η οποία είναι τύπου **Tk( )**. Συνεχίζουμε διαμορφώνοντας το παράθυρο αλλά και το τίτλο του παραθύρου με τις ακόλουθες ρυθμίσεις:

```
root = tk.Tk()
root.title("Python Project")
window_width = 400
window_height = 400
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
x_position = int((screen_width - window_width) / 2)
y_position = int((screen_height - window_height) / 2)
root.geometry(f"{window_width}x{window_height}+{x_position}+{y_position}")
root.configure(borderwidth=5)
root.configure(highlightbackground="#333333", highlightcolor="#333333")

# Set the window title font and size
title_font = ("Helvetica", 13, "bold")
root.option_add("*Font", title_font)
root.option_add("*foreground", "#333333")
bold_font = font.Font(family="Arial", size=12, weight="bold")
```

Ακόμα, θέλουμε να προσθέσουμε και ως εικόνα το logo του Πανεπιστημίου Πατρών. Επομένως, ορίζουμε ένα αντικείμενο **img** τύπου **Image( ),** στο οποίο αποθηκεύουμε την εικόνα που έχουμε τοποθετήσει στο παρακάτω μονοπάτι, την οποία ανοίγουμε καλώντας τη μέθοδο **open( )** πάνω σε αυτό. Στη συνέχεια, αλλάζουμε τις διαστάσεις της εικόνας καλώντας τη μέθοδο **resize( )** πάνω στο **img** και τη μετατρέπουμε στη φωτογραφία **logo** τύπου **TkImage( )** που μπορεί να εμφανιστεί στο GUI μέσω της **PhotoImage()**.

```
img = Image.open(r'C:\Users\user\PycharmProjects\pythonProject1\logo.png')
img = img.resize((300, 110))
logo = ImageTk.PhotoImage(img)
```

Προσθέτουμε ακόμα και κείμενο μέσα στο παράθυρο, το οποίο παρουσιάζει ένα εισαγωγικό μήνυμα και αυτό το καταφέρνουμε με τον εξής τρόπο:

```
text_root = tk.Label(root, text="Εργαστηριακή Άσκηση στην Python\n  Εαρινό Εξάμηνο
2023\n\nΕπεξεργασία Δεδομένων από Αρχείο CSV")
text_root.configure(font=bold_font, foreground="black", background="#f0f0f0")
text_root.pack(padx=10, pady=15)
```

```
image_label = tk.Label(root, image=logo)
image_label.pack(padx=15, pady=15)
```

Παρατηρούμε ότι φτιάχνουμε ένα αντικείμενο **text_root** τύπου **Label( )** πάνω στο **root** και εισάγουμε σε αυτό το κείμενο που επιθυμούμε να εμφανίζεται, εφαρμόζοντάς του και κάποιες ρυθμίσεις, όπως χρώμα, γραμματοσειρά, μέγεθος καθώς και το σημείο του παραθύρου στο οποίο θα βρίσκεται μέσω της **pack( ).**

Εκτός από αυτό, φτιάχνουμε και μια γενική ρύθμιση που θέλουμε να εφαρμοστεί πάνω σε όλα τα κουμπιά του παραθύρου και την ονομάζουμε **button_style**, η οποία περιέχει όλες τις πιο κάτω ρυθμίσεις:

```
button_font = font.Font(family="Helvetica", size=12, weight="bold")
button_style = {
    'foreground': '#ffffff',
    'background': '#20bebe',
    'activeforeground': '#ffffff',
    'activebackground': '#45a049',
    'font': button_font,
    'borderwidth': 2,
    'highlightthickness': 2,
    'relief': 'flat',
}
```

Το μόνο που μας απομένει πλέον για να ολοκληρώσουμε το εισαγωγικό παράθυρο είναι να προσθέσουμε και τα κουμπιά που θα μας επιτρέπουν να δούμε τις γραφικές και τα αρχεία CSV. Για να γίνει αυτό πρέπει να ορίσουμε μια ακόμα μέθοδο με όνομα **popup( )**, η οποία θα εκτελείται όταν ο χρήστης κάνει κλικ στο κουμπί που θα κατασκευάσουμε.

Μέσα στη μέθοδο αυτή ξεκινάμε δημιουργώντας ένα καινούργιο παράθυρο που θα ανοίγει, με όνομα **win**, κάθε φορά που ο χρήστης θα πατάει το κουμπί που θέλουμε να φτιάξουμε. Η δημιουργία του παραθύρου αυτού και η προσθήκη κειμένου γίνεται με τον ίδιο τρόπο που προαναφέραμε, οπότε μεταβαίνουμε κατευθείαν στη δημιουργία των κουμπιών που θα εκτελούν τις λειτουργίες που περιγράφει κάθε ερώτημα. Φτιάχνουμε εννέα κουμπιά της μορφής

buttoni = Button( ) , i = 1,2,3,4,5,6,7,8,9

πάνω στο παράθυρο **win** και εκτός από τις απαραίτητες σχεδιαστικές ρυθμίσεις θέτουμε σε καθένα από ως εντολή η οποία τα ενεργοποιεί τις μεθόδους:

graph_i( ) , i = 1,2,3,4,5,6,7,8 και **export_csv( )**

Έτσι, κάθε φορά που ο χρήστης θα διαλέγει ένα κουμπί θα εμφανίζεται η αντίστοιχη γραφική παράσταση και τα δεδομένα της θα αποθηκεύονται ταυτόχρονα και στη Β.Δ. ή θα εξαχθούν τα αρχεία CSV. Τυπικά φτιάχνουμε και ένα κουμπί εξόδου **close_button**, το οποίο πατώντας το θα μας γυρνάει πίσω στο αρχικό μας παράθυρο.

```
def popup():
    win = Toplevel(root)
    win.title("Graphs")
    win.geometry("1050x1050")

    text = Label(win, text="Graphs")
    button1 = Button(win, text="Profits per Month ($ and Tonnes)", command=lambda: graph_1(),
font="Raleway",
                     bg="#20bebe", fg="white", bd=0, relief="sunken", activebackground="#1b1b1b", height=2,
width=60)
    button2 = Button(win, text="Profits per Country ($ and Tonnes)", command=lambda: graph_2(),
font="Raleway",
                     bg="#20bebe", fg="white", height=2, width=60)
    button3 = Button(win, text="Profits per Transport Mode ($ and Tonnes)", command=lambda: graph_3(),
font="Raleway",
                     bg="#20bebe", fg="white", height=2, width=60)
    button4 = Button(win, text="Profits per Weekday ($ and Tonnes)", command=lambda: graph_4(),
font="Raleway",
                     bg="#20bebe", fg="white", height=2, width=60)
    button5 = Button(win, text="Profits per Commodity ($ and Tonnes)", command=lambda: graph_5(),
font="Raleway",
                     bg="#20bebe", fg="white", height=2, width=60)
    button6 = Button(win, text="Top 5 Months with Biggest Profit ($ and Tonnes)", command=lambda:
graph_6(),
                     font="Raleway", bg="#20bebe", fg="white", height=2, width=60)
    button7 = Button(win, text="Top 5 Commodities with Biggest Profit per Country ($ and Tonnes)",
                     command=lambda: graph_7(), font="Raleway", bg="#20bebe", fg="white", height=2,
width=60)
    button8 = Button(win, text="Weekday with Biggest Profit per Commodity ($ and Tonnes)",
command=lambda: graph_8(),
                     font="Raleway", bg="#20bebe", fg="white", height=2, width=60)
    button9 = Button(win, text="Export Data into CSV Files", command=lambda: export_csv(),
font="Raleway", bg="#20bebe",
                     fg="white", height=2, width=60)
    close_button = Button(win, text="Close", command=lambda: win.destroy(), height=1, width=8)
```

Όσον αφορά τις μορφοποιήσεις αυτών των κουμπιών, για να εφαρμόσουμε σε καθένα από αυτά το φόντο **button_style** που ορίσαμε νωρίτερα, χρησιμοποιούμε την μέθοδο **configure( )** με όρισμα το ****button_style**, καθώς και τη μέθοδο **pack( )** για να τα τοποθετήσουμε μέσα στο παράθυρο. Για να εφαρμοστούν όλα αυτά στο δεύτερο παράθυρο καλούμε πάνω στο αντικείμενο **win** τη μέθοδο **mainloop( )**.

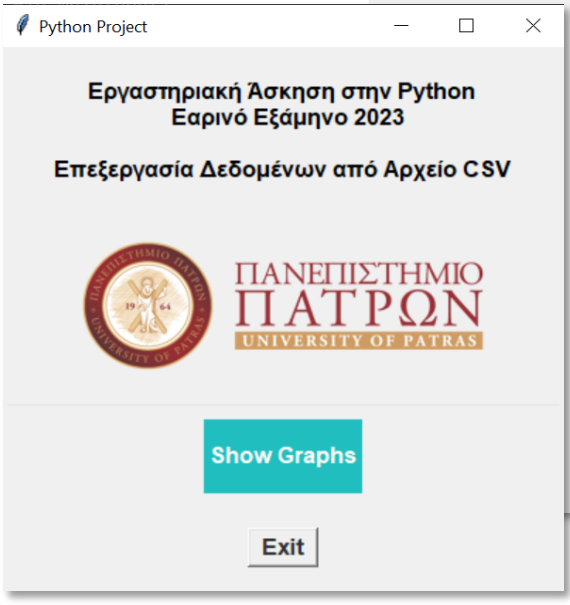
```
    text.pack(padx=0, pady=0)
    button1.configure(**button_style)
    button1.pack(padx=4, pady=4)
    button2.configure(**button_style)
    button2.pack(padx=4, pady=5)
    button3.configure(**button_style)
    button3.pack(padx=4, pady=6)
    button4.configure(**button_style)
    button4.pack(padx=4, pady=7)
    button5.configure(**button_style)
    button5.pack(padx=4, pady=8)
    button6.configure(**button_style)
    button6.pack(padx=5, pady=4)
    button7.configure(**button_style)
    button7.pack(padx=5, pady=5)
```

```
    button8.configure(**button_style)
    button8.pack(padx=5, pady=6)
    button9.configure(**button_style)
    button9.pack(padx=5, pady=7)
    close_button.pack(padx=5, pady=8)

    win.mainloop()
```

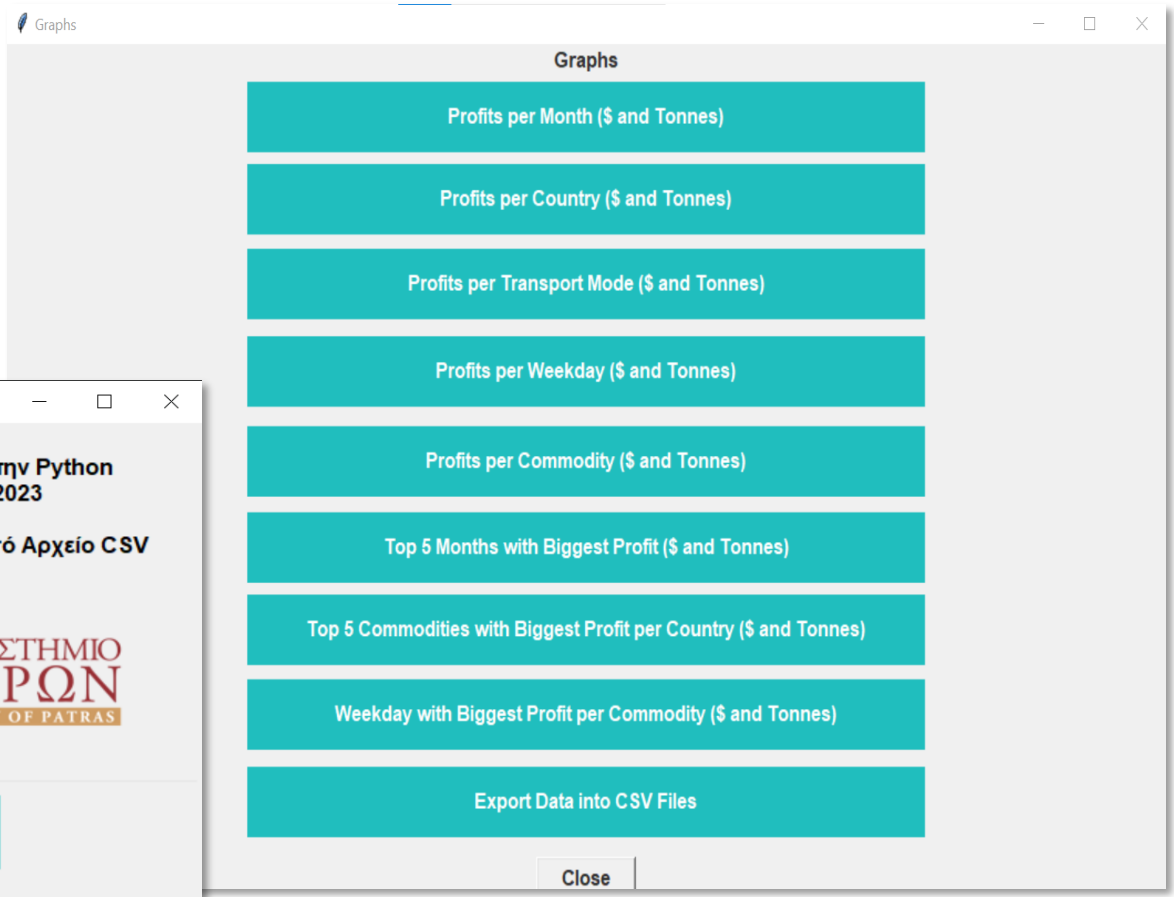
Τελικά, εφόσον έχουμε έτοιμη τη μέθοδο **popup( )** που θα ενεργοποιείται όταν κάνουμε κλικ στο κουμπί του αρχικού παραθύρου, πηγαίνουμε τώρα να το ορίσουμε και αυτό. Φτιάχνουμε λοιπόν ένα αντικείμενο **button** τύπου **Button( )** πάνω στο **root** με εντολή ενεργοποίησης την **popup( )** και ένα άλλο κουμπί **exit_button** το οποίο όταν πατηθεί μέσω της **destroy( )** κλείνει το αρχικό παράθυρο και επομένως όλη τη Διεπαφή. Όμοια, εφαρμόζουμε πάλι στα κουμπία τις ίδιες ρυθμίσεις και για να αποθηκευτούν καλούμε όπως και πριν για τον ίδιο λόγο στο **root** την μέθοδο **mainloop( ).**

```
button = Button(root, text="Show Graphs", command=lambda: popup(), font="Raleway", bg="#20bebe", fg="white", height=2,
                width=10)
button.configure(**button_style)
button.pack(padx=10, pady=10)
exit_button = Button(root, text="Exit", command=lambda: root.destroy(), height=1, width=4)
exit_button.pack(padx=10, pady=15)

root.mainloop()
```

Συνεπώς, μετά από όλα αυτά η Γραφική Διεπαφή μας πρέπει να δείχνει κάπως έτσι:



Παράθυρο 1

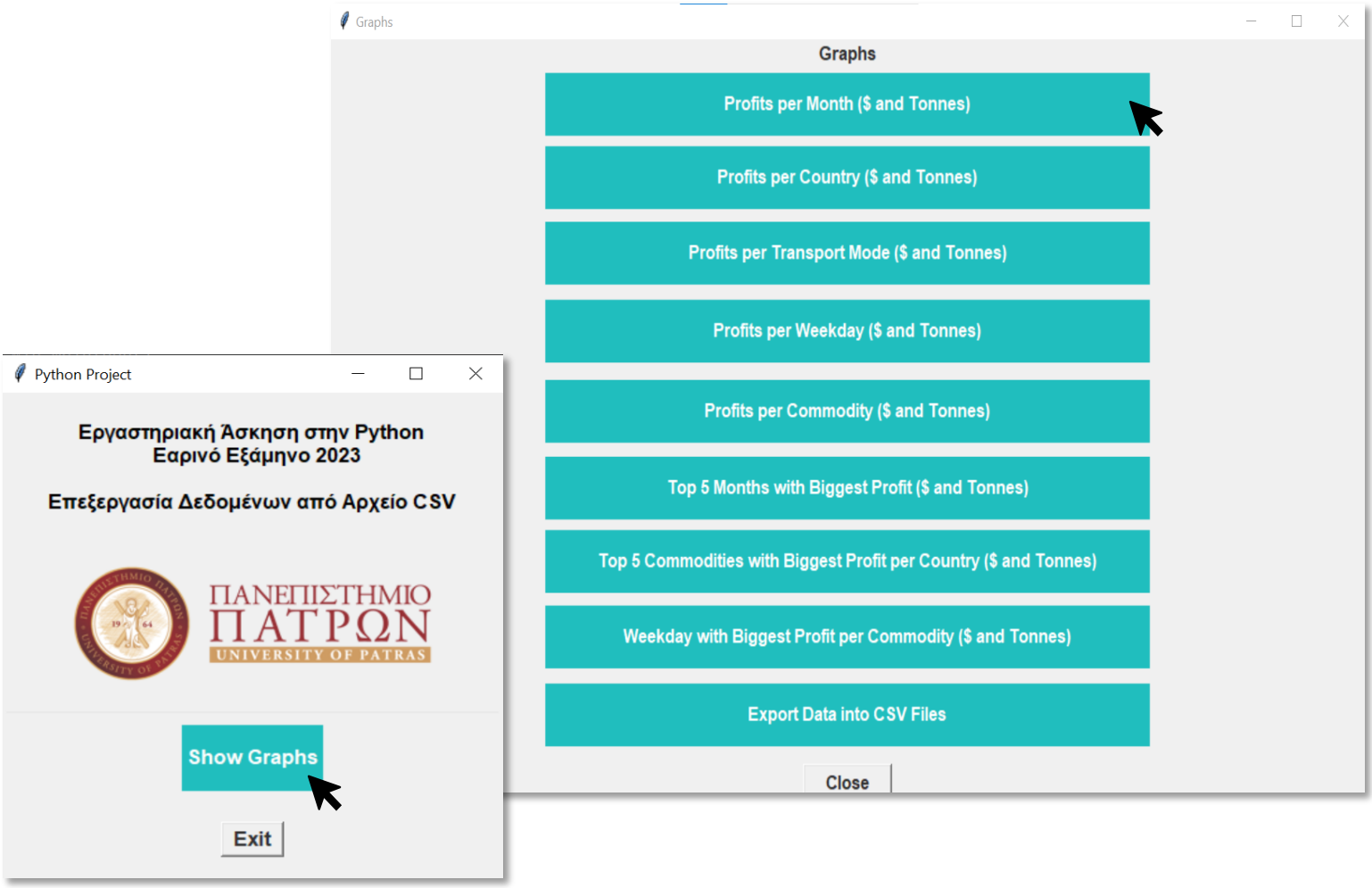


Παράθυρο 2

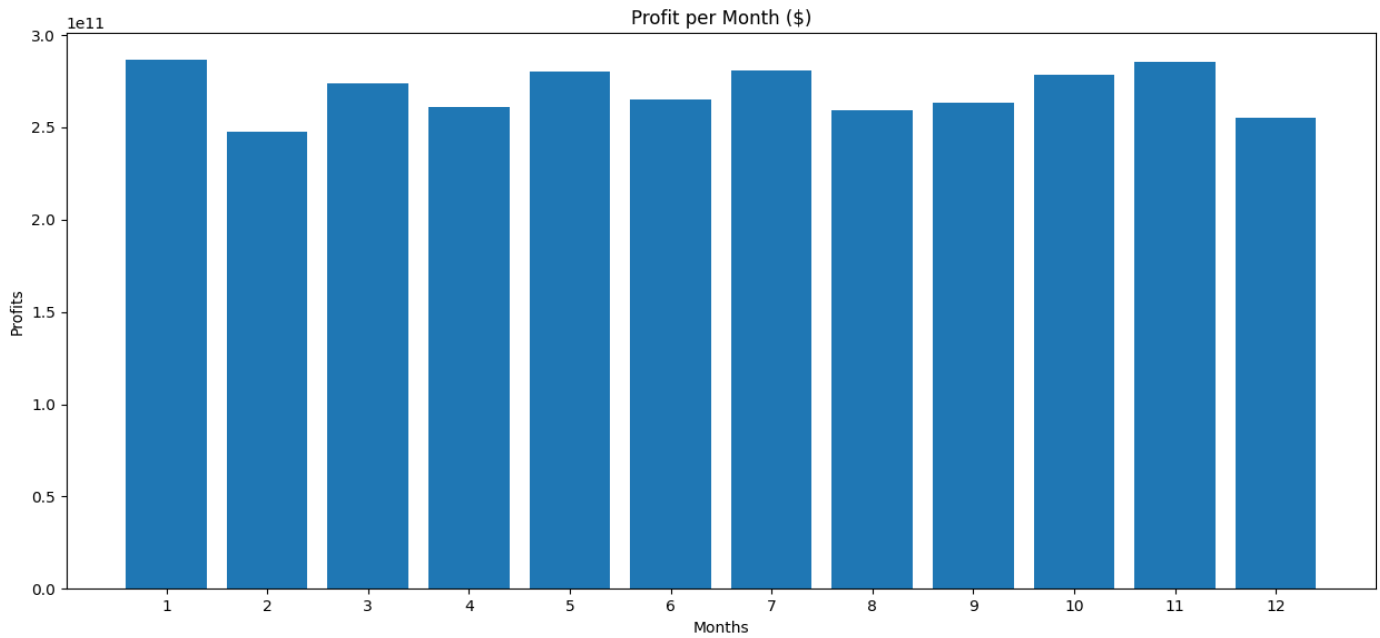
3. Παράρτημα

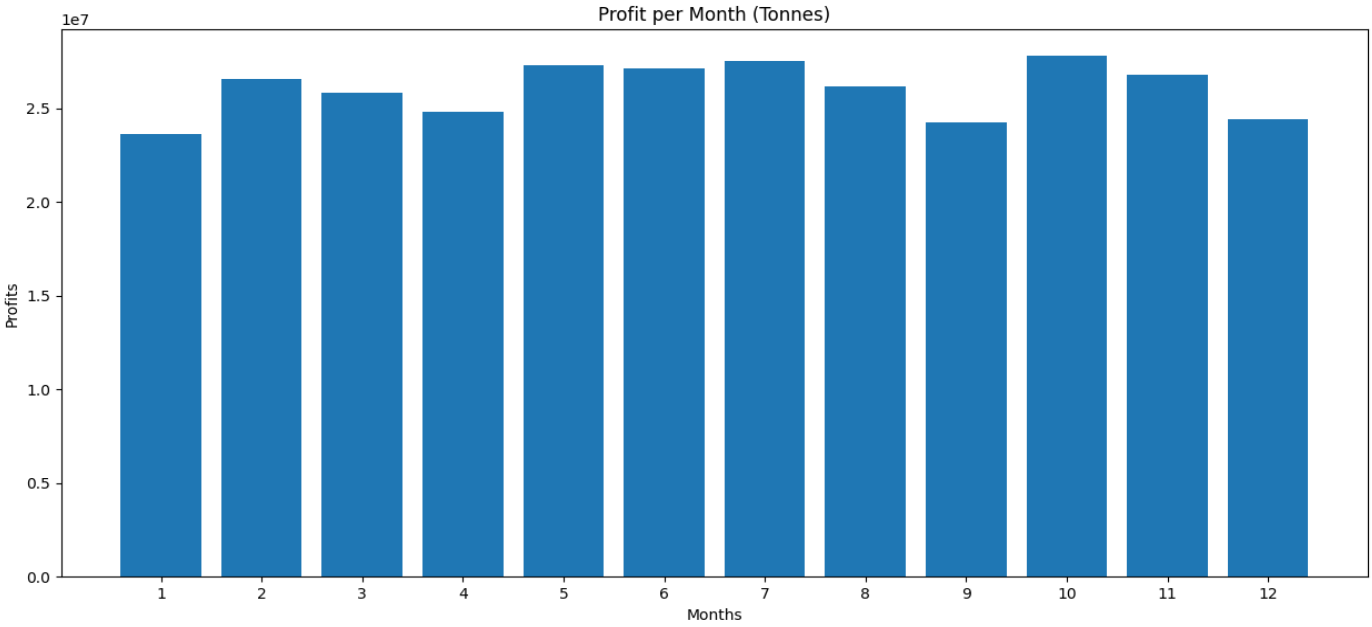
3.1 Σενάριο Χρήσης

Βήμα 1: Μπαίνουμε στη Διεπαφή και επιλέγουμε το κουμπί Show Graphs, από όπου μεταφερόμαστε σε ένα δεύτερο παράθυρο και κάνουμε κλικ στην επιλογή Profits per Month ($ and Tonnes).

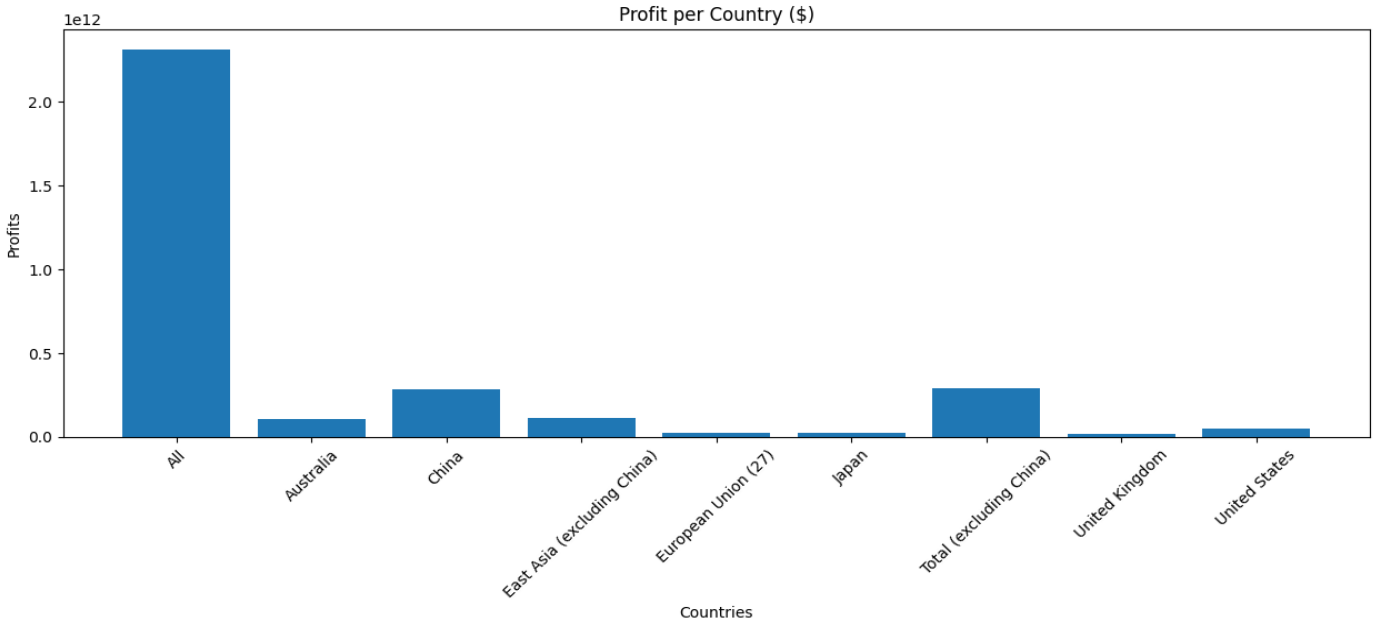


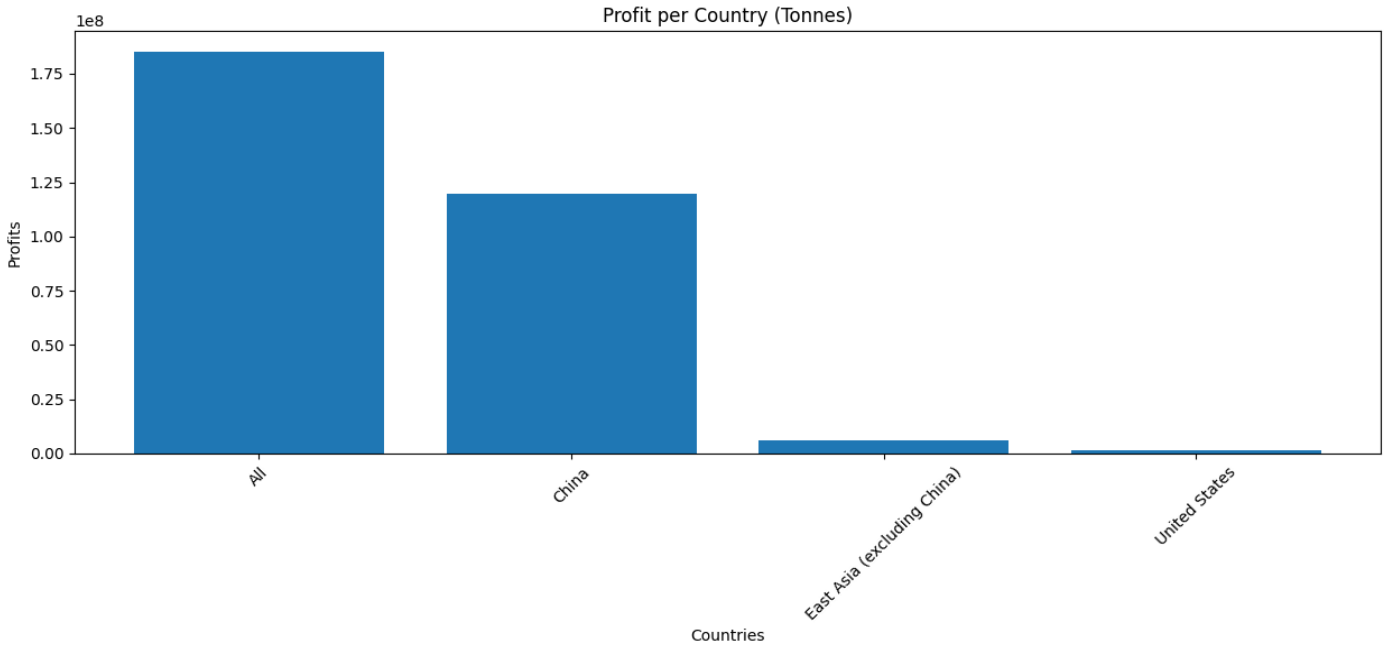
Βήμα 2: Εμφανίζονται οι γραφικές παραστάσεις για το 1ο ερώτημα της άσκησης.



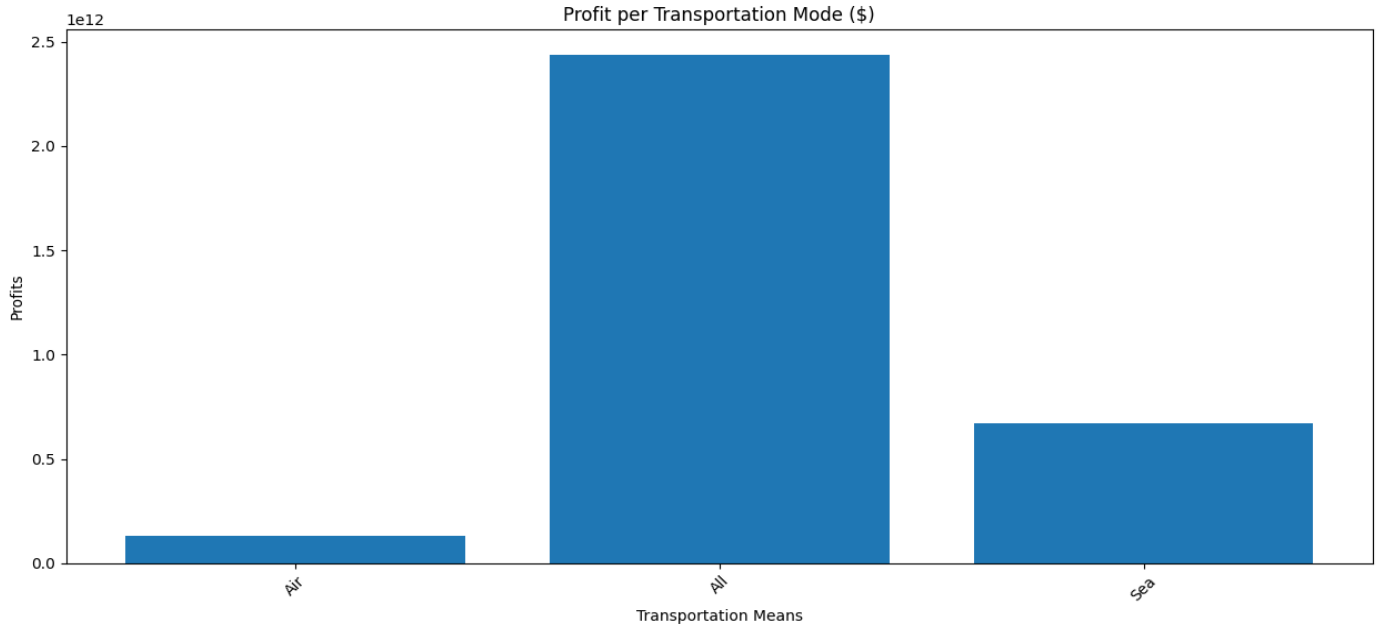


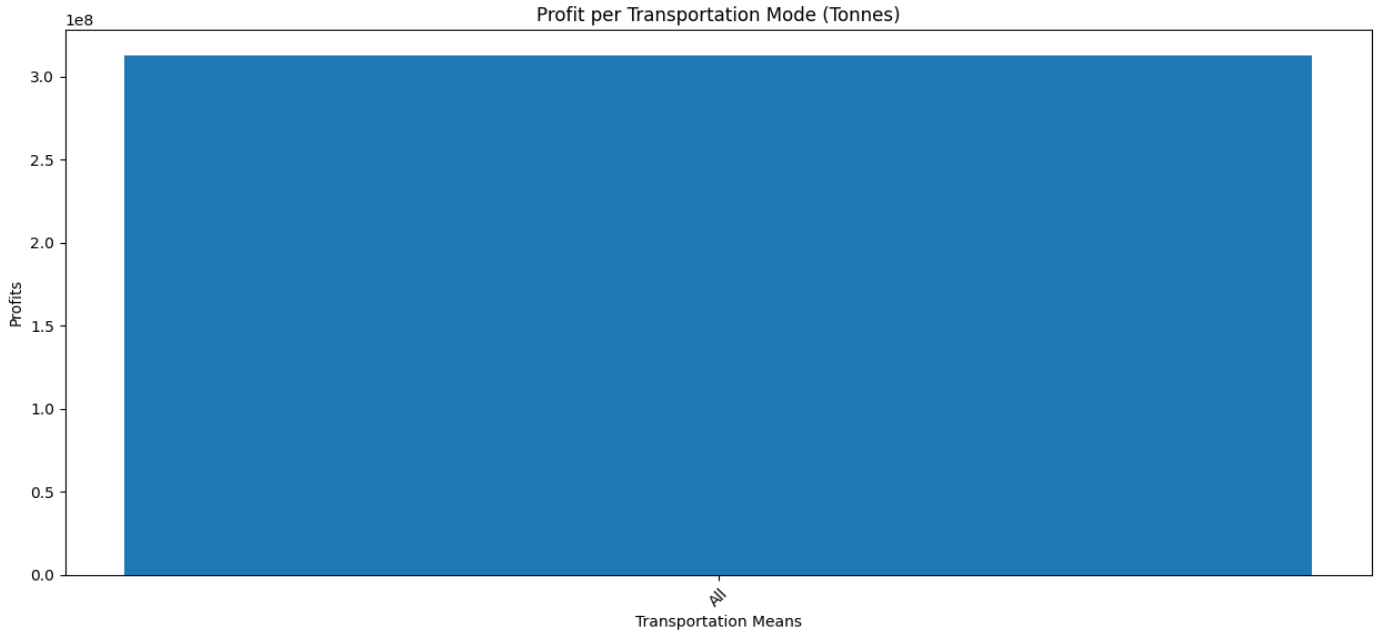
Βήμα 3: Επιλέγουμε το κουμπί Profits per Country ($ and Tonnes) και εμφανίζονται οι γραφικές παραστάσεις για το 2ο ερώτημα της άσκησης.



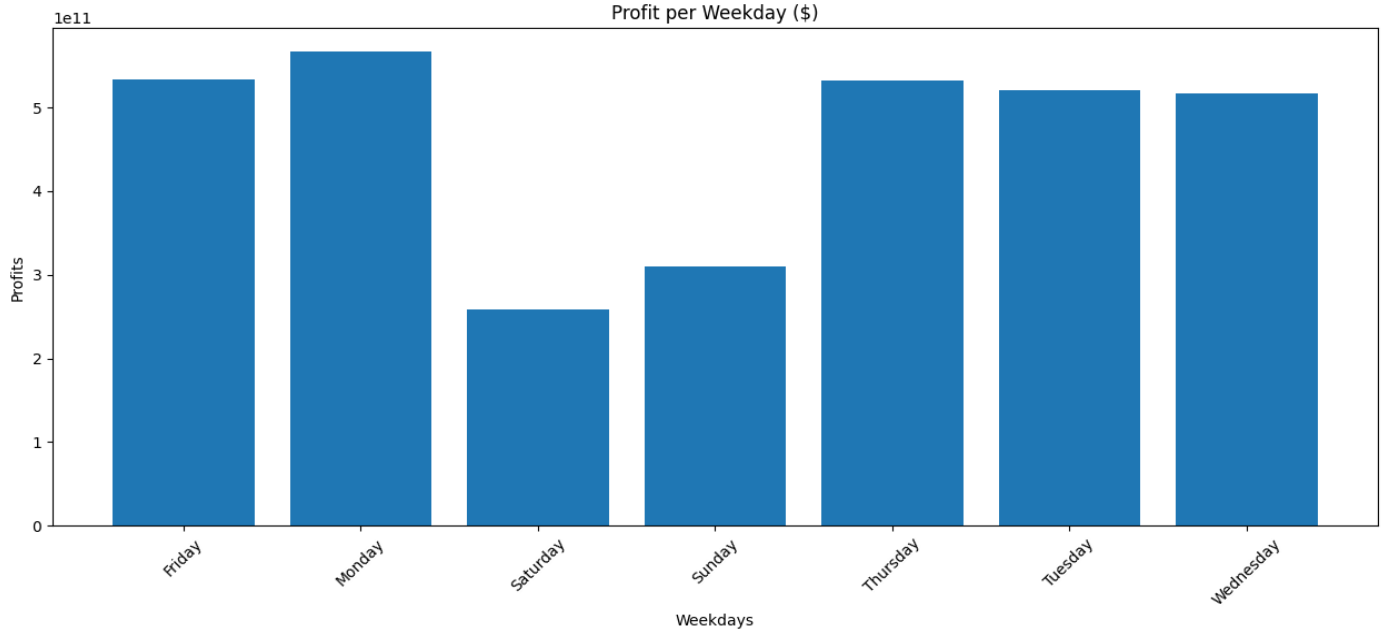


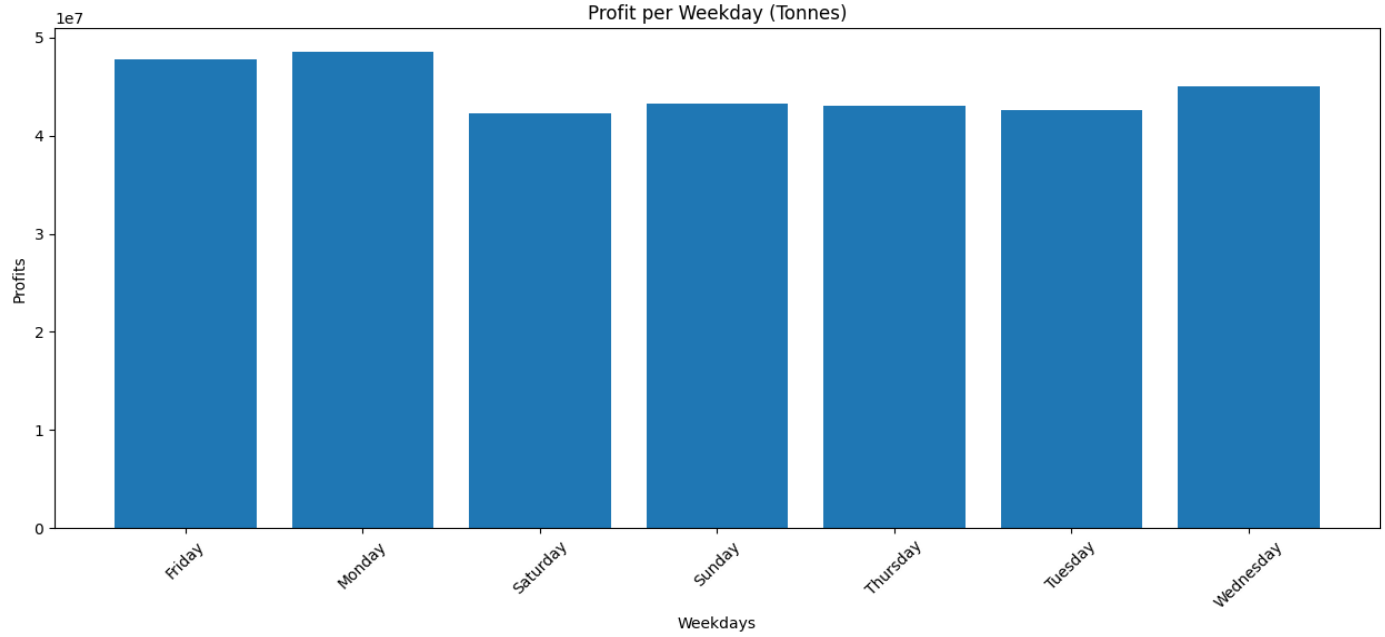
Βήμα 4: Επιλέγουμε το κουμπί Profits per Transport Mode ($ and Tonnes) και εμφανίζονται οι γραφικές παραστάσεις για το 3ο ερώτημα της άσκησης.



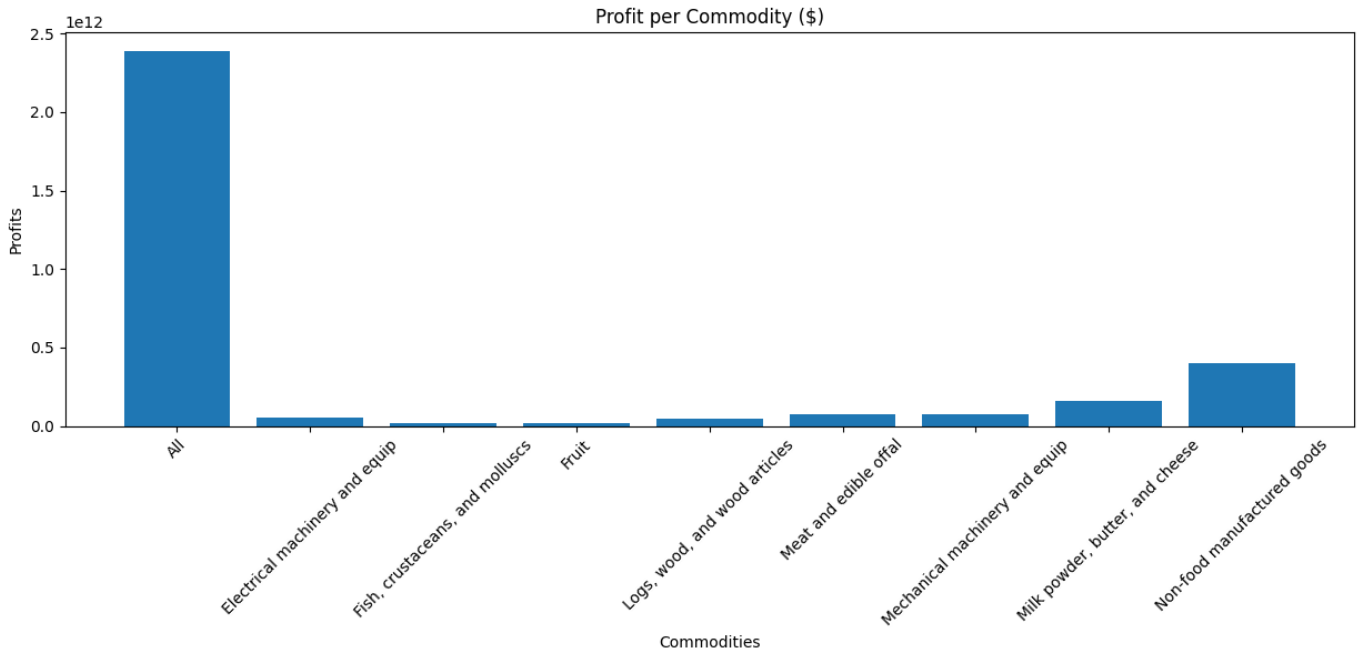


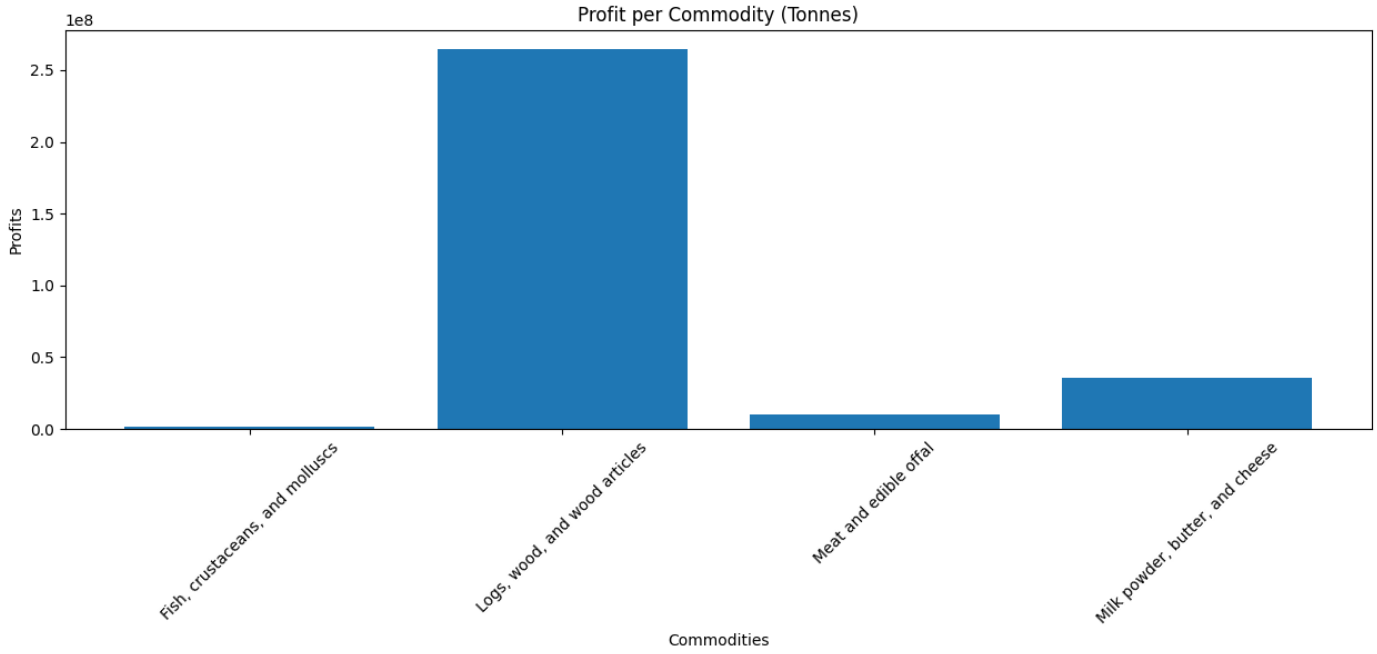
Βήμα 5: Επιλέγουμε το κουμπί Profits per Weekday ($ and Tonnes) και εμφανίζονται οι γραφικές παραστάσεις για το 4ο ερώτημα της άσκησης.



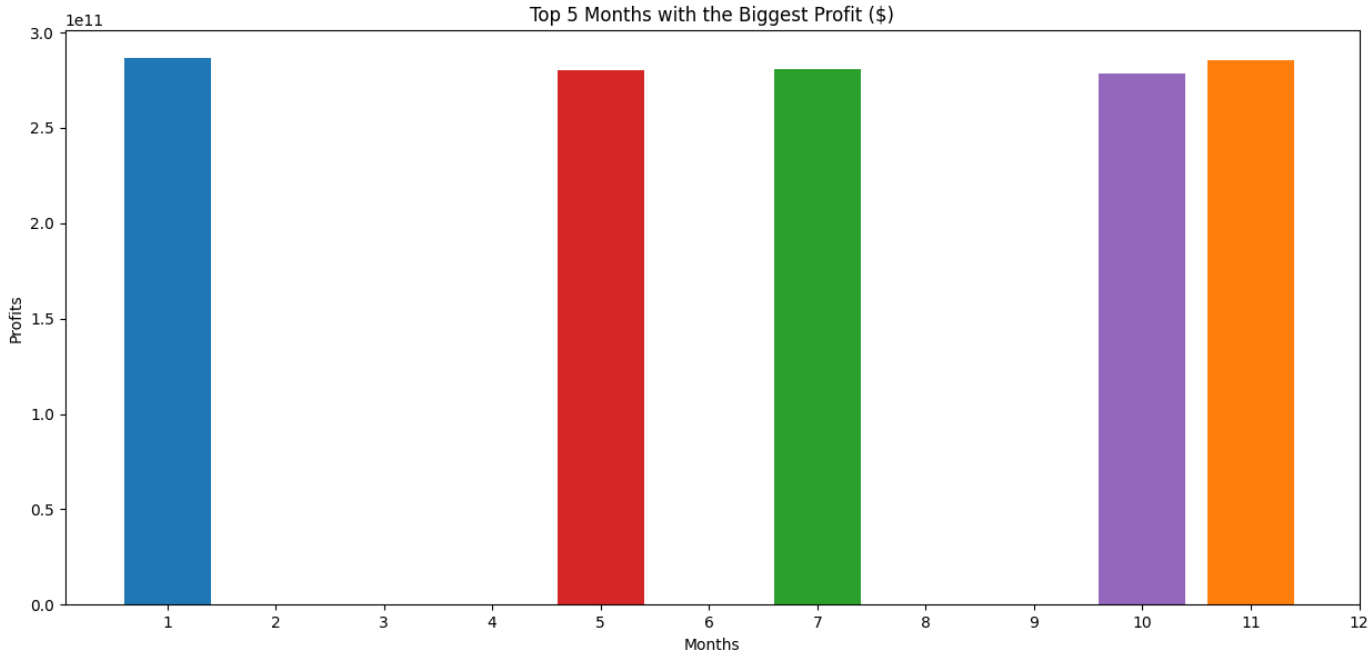


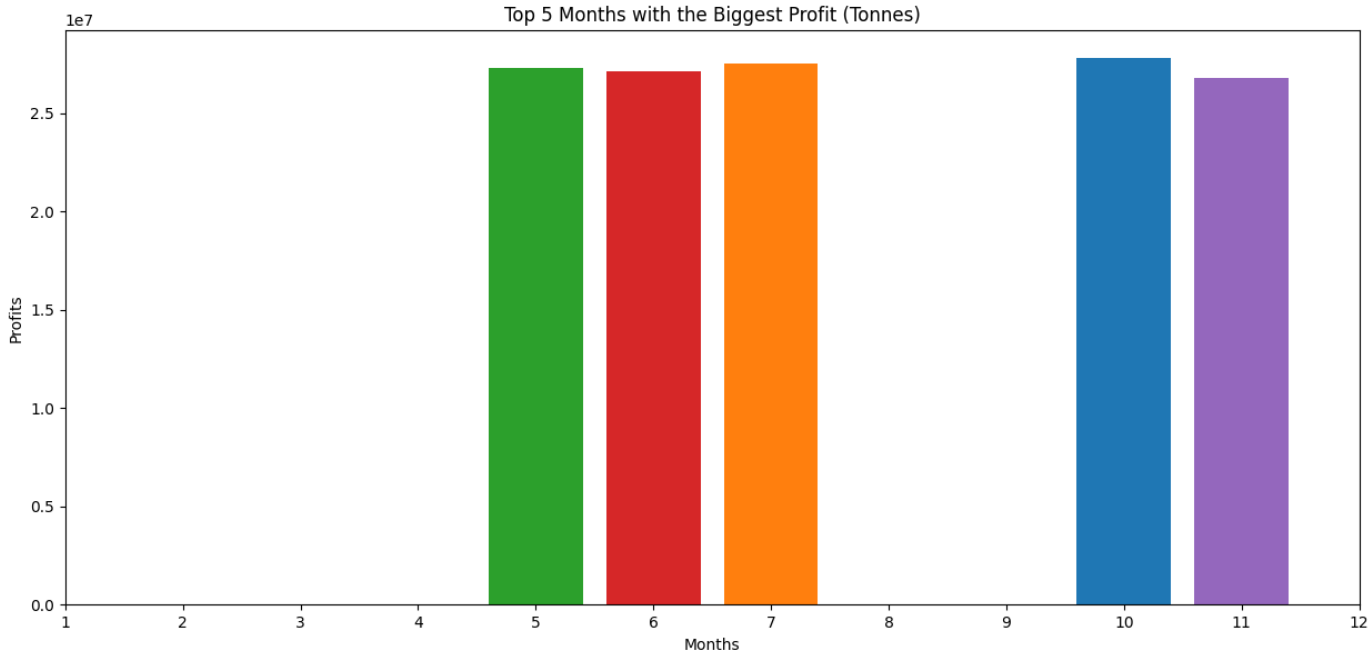
Βήμα 6: Επιλέγουμε το κουμπί Profits per Commodity ($ and Tonnes) και εμφανίζονται οι γραφικές παραστάσεις για το 5ο ερώτημα της άσκησης.



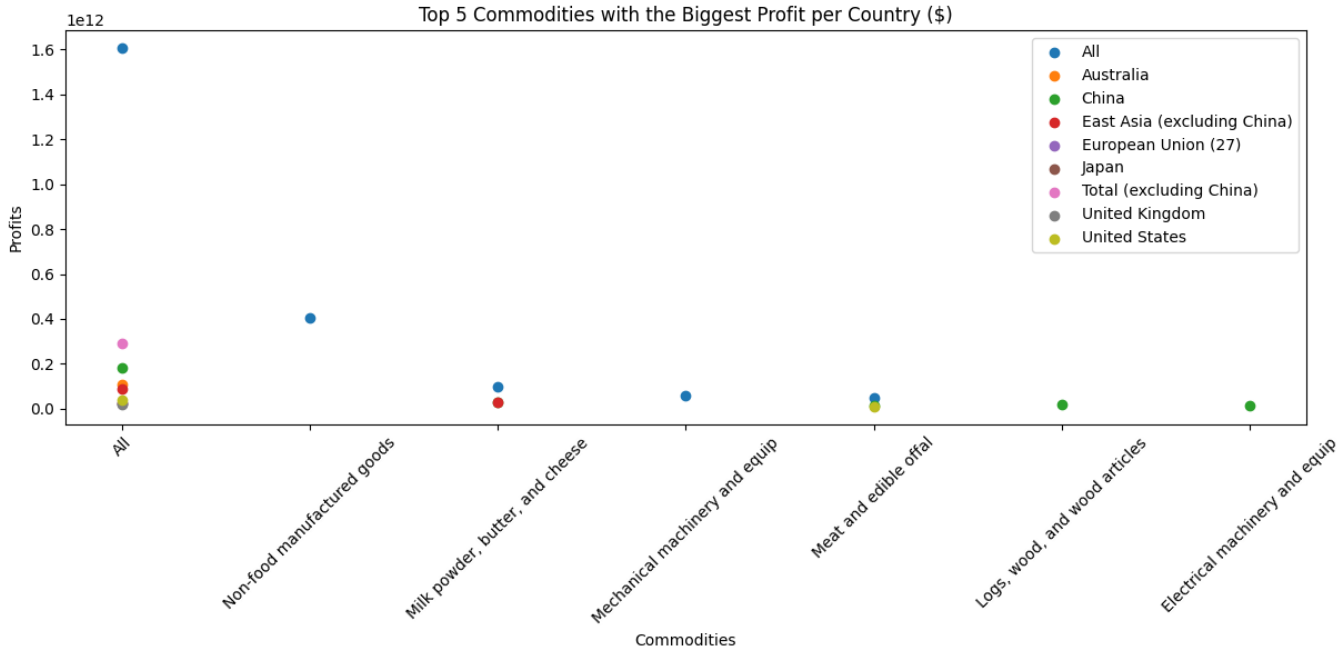


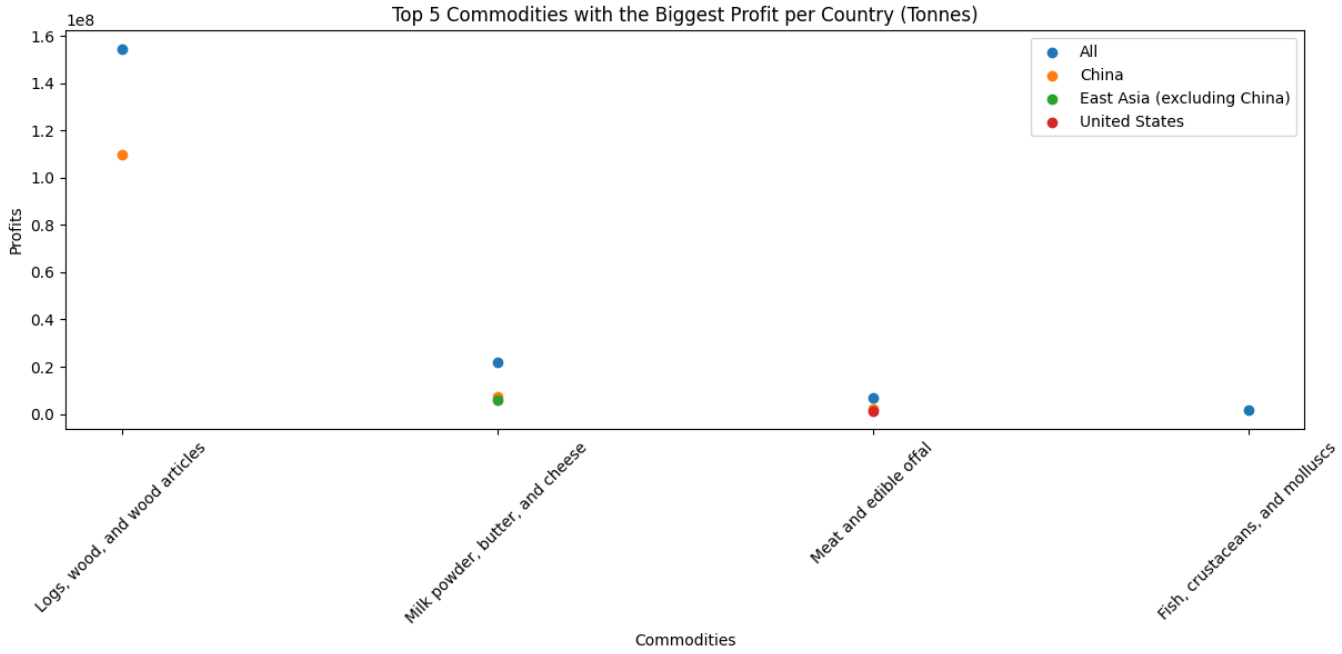
Βήμα 7: Επιλέγουμε το κουμπί Top 5 Months with Biggest Profit ($ and Tonnes) και εμφανίζονται οι γραφικές παραστάσεις για το 6ο ερώτημα της άσκησης.



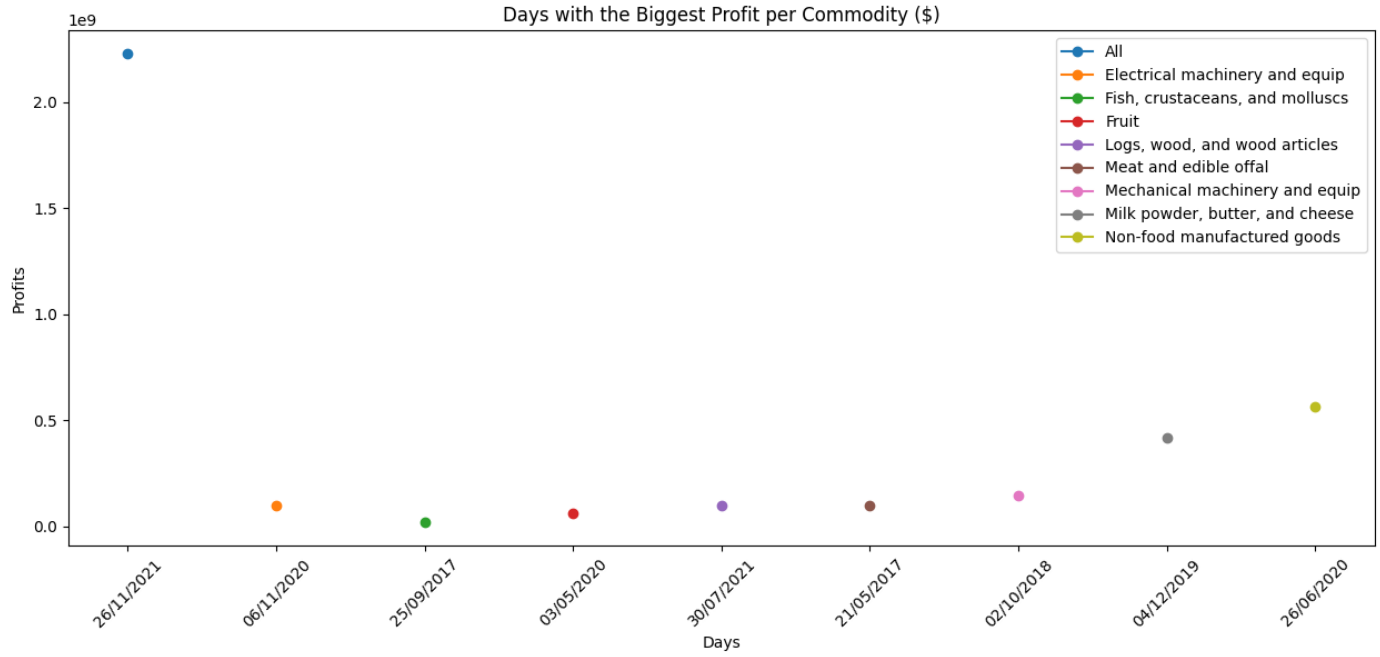


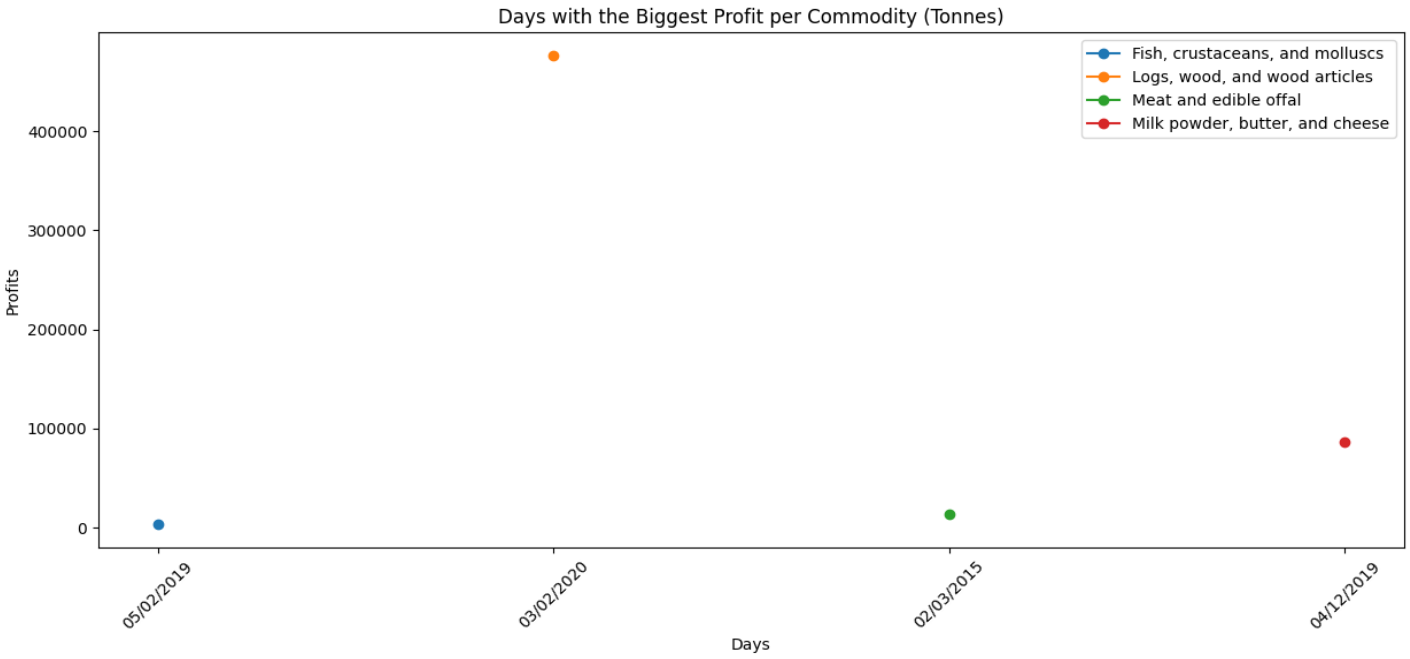
Βήμα 8: Επιλέγουμε το κουμπί Top 5 Commodities with Biggest Profit per Country ($ and Tonnes) και εμφανίζονται οι γραφικές παραστάσεις για το 7ο ερώτημα της άσκησης.



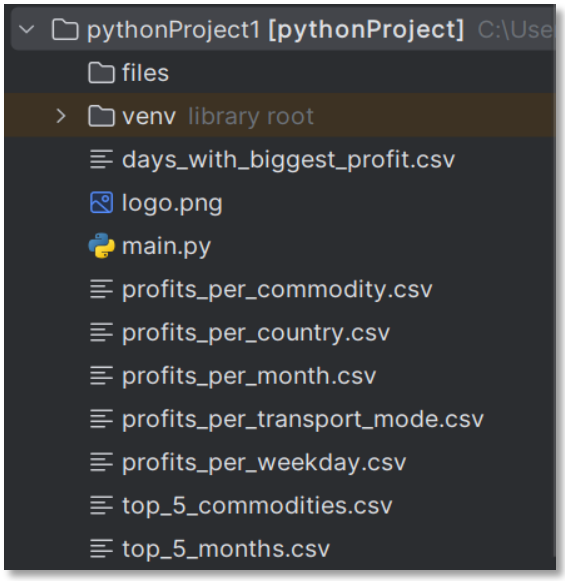


Βήμα 9: Επιλέγουμε το κουμπί Weekday with Biggest Profit per Commodity ($ and Tonnes) και εμφανίζονται οι γραφικές παραστάσεις για το 8ο ερώτημα της άσκησης.

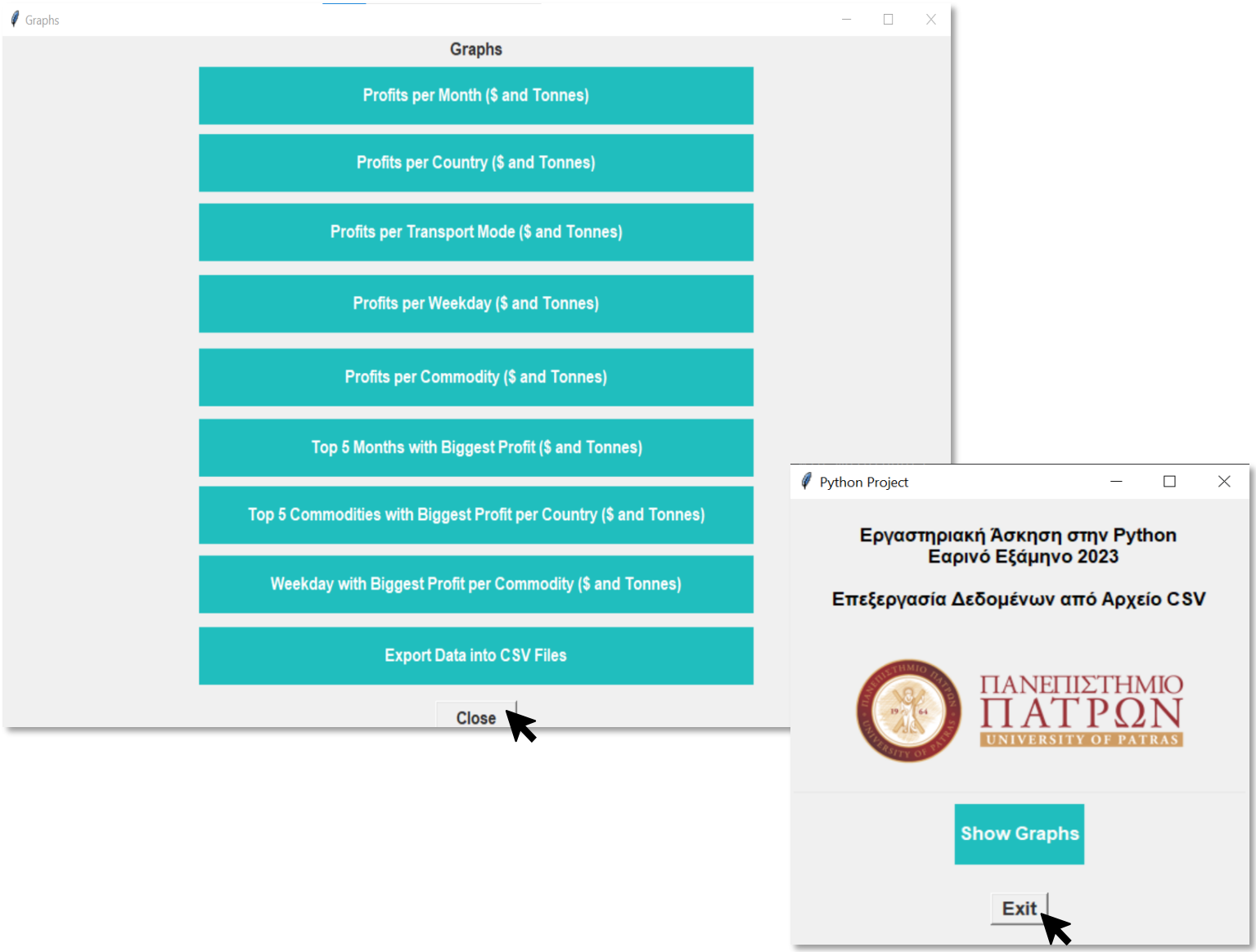




Βήμα 10: Επιλέγουμε το κουμπί Export Data into CSV Files και βλέπουμε ότι τα αρχεία έχουν δημιουργηθεί στο περιβάλλον του PyCharm.



Βήμα 11: Επιλέγουμε το κουμπί Close και επιστρέφουμε στο αρχικό παράθυρο και έπειτα το κουμπί Exit για να εξέλθουμε από τη Διεπαφή.



Βήμα 12: Μεταβαίνουμε στην Επιφάνεια Εργασίας του Workbench της MySQL και βλέπουμε την ολοκληρωμένη Β.Δ., η οποία περιέχει όλους τους πίνακες συμπληρωμένους με τα αντίστοιχα δεδομένα τους.

Profit	Month	Measure
286738000000	1	$
247938000000	2	$
273749000000	3	$
261261000000	4	$
280142000000	5	$
264971000000	6	$
281049000000	7	$
259447000000	8	$
263373000000	9	$
278668000000	10	$
285364000000	11	$
255097000000	12	$
23645000	1	Tonnes
26567000	2	Tonnes
25866000	3	Tonnes
24837000	4	Tonnes
27334000	5	Tonnes

Profit	Country	Measure
2315204000000	All	$
107686000000	Australia	$
282650000000	China	$
116556000000	East Asia (excluding China)	$
26644000000	European Union (27)	$
23155000000	Japan	$
291991000000	Total (excluding China)	$
21591000000	United Kingdom	$
52320000000	United States	$
185349000	All	Tonnes
119573000	China	Tonnes
6137000	East Asia (excluding China)	Tonnes
1338000	United States	Tonnes

Profit	Transport_Mean	Measure
132602000000	Air	$
2436795000000	All	$
668400000000	Sea	$
312397000	All	Tonnes

Profit	Weekday	Measure
533090000000	Friday	$
566712000000	Monday	$
258225000000	Saturday	$
309677000000	Sunday	$
532286000000	Thursday	$
520469000000	Tuesday	$
517338000000	Wednesday	$
47760000	Friday	Tonnes
48527000	Monday	Tonnes
42241000	Saturday	Tonnes
43239000	Sunday	Tonnes
43047000	Thursday	Tonnes
42585000	Tuesday	Tonnes
44998000	Wednesday	Tonnes

Month	Profit	Measure
1	286738000000	$
11	285364000000	$
7	281049000000	$
5	280142000000	$
10	278668000000	$
10	27827000	Tonnes
7	27536000	Tonnes
5	27334000	Tonnes
6	27126000	Tonnes
11	26809000	Tonnes

Profit	Commodity	Measure
2386667000000	All	$
51554000000	Electrical machinery and equip	$
15445000000	Fish, crustaceans, and molluscs	$
22197000000	Fruit	$
50381000000	Logs, wood, and wood articles	$
78512000000	Meat and edible offal	$
72603000000	Mechanical machinery and equip	$
157284000000	Milk powder, butter, and cheese	$
403154000000	Non-food manufactured goods	$
1832000	Fish, crustaceans, and molluscs	Tonnes
264402000	Logs, wood, and wood articles	Tonnes
10372000	Meat and edible offal	Tonnes
35791000	Milk powder, butter, and cheese	Tonnes

Country	Commodity	Profit	Measure
All	All	1603472000000	$
All	Non-food manufactured goods	403154000000	$
All	Milk powder, butter, and cheese	98757000000	$
All	Mechanical machinery and equip	57567000000	$
All	Meat and edible offal	51206000000	$
Australia	All	107686000000	$
China	All	182406000000	$
China	Milk powder, butter, and cheese	31216000000	$
China	Logs, wood, and wood articles	17993000000	$
China	Electrical machinery and equip	16478000000	$
China	Meat and edible offal	15463000000	$
East Asi...	All	89245000000	$
East Asi...	Milk powder, butter, and cheese	27311000000	$
Europe...	All	26644000000	$
Japan	All	23155000000	$
Total (e...	All	291991000000	$
United ...	All	21591000000	$
United ...	Meat and edible offal	11843000000	$
All	Logs, wood, and wood articles	154650000	Tonnes
All	Milk powder, butter, and cheese	22118000	Tonnes
All	Meat and edible offal	6749000	Tonnes
All	Fish, crustaceans, and molluscs	1832000	Tonnes
China	Logs, wood, and wood articles	109752000	Tonnes
China	Milk powder, butter, and cheese	7536000	Tonnes
China	Meat and edible offal	2285000	Tonnes
East Asi...	Milk powder, butter, and cheese	6137000	Tonnes
United ...	Meat and edible offal	1338000	Tonnes

Commodity	Weekday	Profit	Measure
All	26/11/2021	2227000000	$
Electrical machinery and equip	06/11/2020	96000000	$
Fish, crustaceans, and molluscs	25/09/2017	21000000	$
Fruit	03/05/2020	63000000	$
Logs, wood, and wood articles	30/07/2021	97000000	$
Meat and edible offal	21/05/2017	99000000	$
Mechanical machinery and equip	02/10/2018	143000000	$
Milk powder, butter, and cheese	04/12/2019	419000000	$
Non-food manufactured goods	26/06/2020	565000000	$
Fish, crustaceans, and molluscs	05/02/2019	4000	Tonnes
Logs, wood, and wood articles	03/02/2020	476000	Tonnes
Meat and edible offal	02/03/2015	14000	Tonnes
Milk powder, butter, and cheese	04/12/2019	87000	Tonnes

3.2 Κώδικας

```
import pandas as pd
import matplotlib.pyplot as plt
import tkinter as tk
from tkinter import *
from tkinter import ttk
from tkinter import font
from PIL import Image, ImageTk
import mysql.connector

root = tk.Tk()
root.title("Python Project")
window_width = 400
window_height = 400
screen_width = root.winfo_screenwidth()
screen_height = root.winfo_screenheight()
x_position = int((screen_width - window_width) / 2)
y_position = int((screen_height - window_height) / 2)
root.geometry(f"{window_width}x{window_height}+{x_position}+{y_position}")
root.configure(borderwidth=5)
root.configure(highlightbackground="#333333", highlightcolor="#333333")

# Set the window title font and size
title_font = ("Helvetica", 13, "bold")
root.option_add("*Font", title_font)
root.option_add("*foreground", "#333333")
bold_font = font.Font(family="Arial", size=12, weight="bold")

img = Image.open(r'C:\Users\user\PycharmProjects\pythonProject1\logo.png')
img = img.resize((300, 110))
logo = ImageTk.PhotoImage(img)

text_root = tk.Label(root, text="Εργαστηριακή Άσκηση στην Python\n  Εαρινό Εξάμηνο
2023\n\nΕπεξεργασία Δεδομένων από Αρχείο CSV")
text_root.configure(font=bold_font, foreground="black", background="#f0f0f0")
text_root.pack(padx=10, pady=15)
image_label = tk.Label(root, image=logo)
image_label.pack(padx=15, pady=15)

button_font = font.Font(family="Helvetica", size=12, weight="bold")
button_style = {
    'foreground': '#ffffff',
    'background': '#20bebe',
    'activeforeground': '#ffffff',
    'activebackground': '#45a049',
    'font': button_font,
    'borderwidth': 2,
    'highlightthickness': 2,
    'relief': 'flat',
}

style = ttk.Style()
style.configure("Custom.TFrame", background="#e6e6e6")
frame = ttk.Frame(root, style="Custom.TFrame")
frame.pack(fill=tk.BOTH, expand=True)
```

```
mydb = mysql.connector.connect(host="localhost", user="root", passwd="ju59GSX4yn",
database="project")
cursor = mydb.cursor()

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'
data = pd.read_csv(url)

def graph_1():
    # Μετατρέπουμε το αλφαριθμητικό που αντιπροσωπεύει την ημερομηνία στη μορφή datetime
    # με format=mixed γιατί δεν είναι της μορφής day-month-year
    data['Date'] = pd.to_datetime(data['Date'], format='mixed')

    # Ομαδοποιούμε τα δεδομένα βάσει μήνα και υπολογίζουμε το τζίρο για κάθε μήνα και measure
    grouped = data.groupby(['Measure', data['Date'].dt.month])['Value'].sum().reset_index()

    # Βρίσκουμε τις διακριτές τιμές του measure
    unique_measures = grouped['Measure'].unique()

    # Διαβάζουμε τα δεδομένα για κάθε διαφορετική τιμή του measure
    for measure in unique_measures:
        # Ομαδοποιούμε τα δεδομένα για κάθε διαφορετική τιμή του measure
        measure_data = grouped[grouped['Measure'] == measure]

        for index, row in measure_data.iterrows():
            month = row['Date']
            value = row['Value']
            cursor.execute("INSERT INTO profit_per_month (Profit, Month, Measure) VALUES (%s, %s, %s)",
                           (value, month, measure))

        # Εισάγουμε τις αλλαγές που πραγματοποιήσαμε στον cursor τη ΒΔ
        mydb.commit()

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        plt.bar(measure_data['Date'], measure_data['Value'])
        plt.title(f"Profit per Month ({measure})")
        plt.xlabel("Months")
        plt.ylabel("Profits")
        plt.xticks(measure_data['Date'])
        plt.tight_layout()
        plt.show()

def graph_2():
    grouped = data.groupby(['Country', 'Measure'])['Value'].sum().reset_index()
    unique_measures = grouped['Measure'].unique()

    for measure in unique_measures:
        measure_data = grouped[grouped['Measure'] == measure]

        for index, row in measure_data.iterrows():
            country = row['Country']
            value = row['Value']
            cursor.execute("INSERT INTO profit_per_country (Profit, Country, Measure) VALUES (%s, %s, %s)",
                           (value, country, measure))

        # Εισάγουμε τις αλλαγές που πραγματοποιήσαμε στον cursor της ΒΔ και κλείνει η σύνδεση
```

```
        mydb.commit()

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        plt.bar(measure_data['Country'], measure_data['Value'])
        plt.title(f"Profit per Country ({measure})")
        plt.xlabel("Countries")
        plt.ylabel("Profits")
        plt.xticks(measure_data['Country'], rotation=45)
        plt.tight_layout()
        plt.show()

def graph_3():
    grouped = data.groupby(['Transport_Mode', 'Measure'])['Value'].sum().reset_index()
    unique_measures = grouped['Measure'].unique()

    for measure in unique_measures:
        measure_data = grouped[grouped['Measure'] == measure]

        for index, row in measure_data.iterrows():
            transport_mean = row['Transport_Mode']
            value = row['Value']
            cursor.execute(
                "INSERT INTO profit_per_transport_mean (Profit, Transport_Mean, Measure) VALUES (%s, %s, %s)",
                (value, transport_mean, measure))

        # Εισάγουμε τις αλλαγές που πραγματοποιήσαμε στον cursor της ΒΔ και κλείνει η σύνδεση
        mydb.commit()

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        plt.bar(measure_data['Transport_Mode'], measure_data['Value'])
        plt.title(f"Profit per Transportation Mode ({measure})")
        plt.xlabel("Transportation Means")
        plt.ylabel("Profits")
        plt.xticks(measure_data['Transport_Mode'], rotation=45)
        plt.tight_layout()
        plt.show()

def graph_4():
    grouped = data.groupby(['Weekday', 'Measure'])['Value'].sum().reset_index()
    unique_measures = grouped['Measure'].unique()

    for measure in unique_measures:
        measure_data = grouped[grouped['Measure'] == measure]

        for index, row in measure_data.iterrows():
            weekday = row['Weekday']
            value = row['Value']
            cursor.execute("INSERT INTO profit_per_weekday (Profit, Weekday, Measure) VALUES (%s, %s, %s)",
                           (value, weekday, measure))

        # Εισάγουμε τις αλλαγές που πραγματοποιήσαμε στον cursor της ΒΔ και κλείνει η σύνδεση
        mydb.commit()

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        plt.bar(measure_data['Weekday'], measure_data['Value'])
```

```
        plt.title(f"Profit per Weekday ({measure})")
        plt.xlabel("Weekdays")
        plt.ylabel("Profits")
        plt.xticks(measure_data['Weekday'], rotation=45)
        plt.tight_layout()
        plt.show()

def graph_5():
    grouped = data.groupby(['Commodity', 'Measure'])['Value'].sum().reset_index()
    unique_measures = grouped['Measure'].unique()

    for measure in unique_measures:
        measure_data = grouped[grouped['Measure'] == measure]

        for index, row in measure_data.iterrows():
            commodity = row['Commodity']
            value = row['Value']
            cursor.execute("INSERT INTO profit_per_commodity (Profit, Commodity, Measure) VALUES (%s, %s,
%s)",
                           (value, commodity, measure))

        # Εισάγουμε τις αλλαγές που πραγματοποιήσαμε στον cursor της ΒΔ και κλείνει η σύνδεση
        mydb.commit()

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        plt.bar(measure_data['Commodity'], measure_data['Value'])
        plt.title(f"Profit per Commodity ({measure})")
        plt.xlabel("Commodities")
        plt.ylabel("Profits")
        plt.xticks(measure_data['Commodity'], rotation=45)
        plt.tight_layout()
        plt.show()

def graph_6():
    data = pd.read_csv(url)
    data['Date'] = pd.to_datetime(data['Date'], format='mixed')
    grouped = data.groupby(['Measure', data['Date'].dt.month])['Value'].sum().reset_index()

    unique_measures = grouped['Measure'].unique()

    for measure in unique_measures:
        measure_data = grouped[grouped['Measure'] == measure]

        # Find the top 5 months for each measure
        top_5_months = (
        measure_data.groupby('Measure')
        .apply(lambda x: x.reset_index(drop=True).nlargest(5, 'Value'))
        .reset_index(drop=True)
        )

        for _, row in top_5_months.iterrows():
            query = 'INSERT INTO top_5_months (Month, Profit, Measure) VALUES (%s, %s, %s)'
            cursor.execute(query, (row['Date'], row['Value'], measure))
        # Εισάγουμε τις αλλαγές που πραγματοποιήσαμε στον cursor της ΒΔ και κλείνει η σύνδεση
        mydb.commit()

        # Σχεδιάζουμε τη γραφική παράσταση
```

```
        plt.figure(figsize=(10, 6))
        for _, data in top_5_months.iterrows():
            plt.bar(data['Date'], data['Value'])

        plt.title(f"Top 5 Months with the Biggest Profit ({measure})")
        plt.xlabel("Months")
        plt.ylabel("Profits")
        plt.xticks(measure_data['Date'])
        plt.tight_layout()
    plt.show()

def graph_7():
    data = pd.read_csv(url)
    grouped = data.groupby(['Measure', 'Country', 'Commodity'])['Value'].sum().reset_index()

    unique_measures = grouped['Measure'].unique()

    for measure in unique_measures:
        measure_data = grouped[grouped['Measure'] == measure]

        # Find the top 5 commodities with the largest sum per country for each measure
        top_5_commodities = (
            measure_data.groupby(['Measure', 'Country'])
            .apply(lambda x: x.nlargest(5, 'Value'))
            .reset_index(drop=True)
        )

        for _, row in top_5_commodities.iterrows():
            query = 'INSERT INTO top_5_commodities (Country, Commodity, Profit, Measure) VALUES (%s, %s,
%s, %s)'
            cursor.execute(query, (row['Country'], row['Commodity'], row['Value'], measure))
        # Εισάγουμε τις αλλαγές που πραγματοποιήσαμε στον cursor της ΒΔ και κλείνει η σύνδεση
        mydb.commit()

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        for country, data in top_5_commodities.groupby('Country'):
            plt.scatter(data['Commodity'], data['Value'], marker='o', label=f"{country}")

        plt.title(f"Top 5 Commodities with the Biggest Profit per Country ({measure})")
        plt.xlabel("Commodities")
        plt.ylabel("Profits")
        plt.xticks(rotation=45)
        plt.legend()
        plt.tight_layout()
    plt.show()

def graph_8():
    data = pd.read_csv(url)
    grouped = data.groupby(['Measure', 'Commodity', 'Date'])['Value'].sum().reset_index()

    # Find the unique values in the Measure column
    unique_measures = grouped['Measure'].unique()

    # Create separate plots for each value in the Measure column
    for measure in unique_measures:
        # Filter the data for the specific value of the Measure column
        measure_data = grouped[grouped['Measure'] == measure]
```

```
        # Find the day with the biggest sum of values for each commodity
        max_days = measure_data.groupby('Commodity')['Value'].idxmax()
        max_days_data = measure_data.loc[max_days]

        for _, row in max_days_data.iterrows():
            query = 'INSERT INTO days_with_biggest_profit (Commodity, Weekday, Profit, Measure) VALUES
(%s, %s, %s, %s)'
            cursor.execute(query, (row['Commodity'], row['Date'], row['Value'], measure))
        mydb.commit()

        # Σχεδιάζουμε τη γραφική παράσταση
        plt.figure(figsize=(10, 6))
        for commodity, data in max_days_data.groupby('Commodity'):
            plt.plot(data['Date'], data['Value'], marker='o', label=f"{commodity}")

        plt.title(f"Days with the Biggest Profit per Commodity ({measure})")
        plt.xlabel("Days")
        plt.ylabel("Profits")
        plt.xticks(rotation=45)
        plt.legend()
        plt.tight_layout()
    plt.show()

def export_csv():
    query = "SELECT * FROM profit_per_month"
    cursor.execute(query)
    res = cursor.fetchall()

    for row in res:
        new_data = pd.read_sql_query(query, mydb)
        new_data.to_csv('profits_per_month.csv', index=FALSE)

    query = "SELECT * FROM profit_per_country"
    cursor.execute(query)
    res = cursor.fetchall()

    for row in res:
        new_data = pd.read_sql_query(query, mydb)
        new_data.to_csv('profits_per_country.csv', index=FALSE)

    query = "SELECT * FROM profit_per_transport_mean"
    cursor.execute(query)
    res = cursor.fetchall()

    for row in res:
        new_data = pd.read_sql_query(query, mydb)
        new_data.to_csv('profits_per_transport_mode.csv', index=FALSE)

    query = "SELECT * FROM profit_per_weekday"
    cursor.execute(query)
    res = cursor.fetchall()

    for row in res:
        new_data = pd.read_sql_query(query, mydb)
        new_data.to_csv('profits_per_weekday.csv', index=FALSE)

    query = "SELECT * FROM profit_per_commodity"
```

```
    cursor.execute(query)
    res = cursor.fetchall()

    for row in res:
        new_data = pd.read_sql_query(query, mydb)
        new_data.to_csv('profits_per_commodity.csv', index=FALSE)

    query = "SELECT * FROM top_5_months"
    cursor.execute(query)
    res = cursor.fetchall()

    for row in res:
        new_data = pd.read_sql_query(query, mydb)
        new_data.to_csv('top_5_months.csv', index=FALSE)

    query = "SELECT * FROM top_5_commodities"
    cursor.execute(query)
    res = cursor.fetchall()

    for row in res:
        new_data = pd.read_sql_query(query, mydb)
        new_data.to_csv('top_5_commodities.csv', index=FALSE)

    query = "SELECT * FROM days_with_biggest_profit"
    cursor.execute(query)
    res = cursor.fetchall()

    for row in res:
        new_data = pd.read_sql_query(query, mydb)
        new_data.to_csv('days_with_biggest_profit.csv', index=FALSE)

def popup():
    win = Toplevel(root)
    win.title("Graphs")
    win.geometry("1050x1050")

    text = Label(win, text="Graphs")
    button1 = Button(win, text="Profits per Month ($ and Tonnes)", command=lambda: graph_1(),
font="Raleway",
                     bg="#20bebe", fg="white", bd=0, relief="sunken", activebackground="#1b1b1b", height=2,
width=60)
    button2 = Button(win, text="Profits per Country ($ and Tonnes)", command=lambda: graph_2(),
font="Raleway",
                     bg="#20bebe", fg="white", height=2, width=60)
    button3 = Button(win, text="Profits per Transport Mode ($ and Tonnes)", command=lambda: graph_3(),
font="Raleway",
                     bg="#20bebe", fg="white", height=2, width=60)
    button4 = Button(win, text="Profits per Weekday ($ and Tonnes)", command=lambda: graph_4(),
font="Raleway",
                     bg="#20bebe", fg="white", height=2, width=60)
    button5 = Button(win, text="Profits per Commodity ($ and Tonnes)", command=lambda: graph_5(),
font="Raleway",
                     bg="#20bebe", fg="white", height=2, width=60)
    button6 = Button(win, text="Top 5 Months with Biggest Profit ($ and Tonnes)", command=lambda:
graph_6(),
                     font="Raleway", bg="#20bebe", fg="white", height=2, width=60)
    button7 = Button(win, text="Top 5 Commodities with Biggest Profit per Country ($ and Tonnes)",
                     command=lambda: graph_7(), font="Raleway", bg="#20bebe", fg="white", height=2,
```

```
width=60)
    button8 = Button(win, text="Weekday with Biggest Profit per Commodity ($ and Tonnes)",
command=lambda: graph_8(),
                     font="Raleway", bg="#20bebe", fg="white", height=2, width=60)
    button9 = Button(win, text="Export Data into CSV Files", command=lambda: export_csv(),
font="Raleway", bg="#20bebe",
                     fg="white", height=2, width=60)
    close_button = Button(win, text="Close", command=lambda: win.destroy(), height=1, width=8)

    text.pack(padx=0, pady=0)
    button1.configure(**button_style)
    button1.pack(padx=4, pady=4)
    button2.configure(**button_style)
    button2.pack(padx=4, pady=5)
    button3.configure(**button_style)
    button3.pack(padx=4, pady=6)
    button4.configure(**button_style)
    button4.pack(padx=4, pady=7)
    button5.configure(**button_style)
    button5.pack(padx=4, pady=8)
    button6.configure(**button_style)
    button6.pack(padx=5, pady=4)
    button7.configure(**button_style)
    button7.pack(padx=5, pady=5)
    button8.configure(**button_style)
    button8.pack(padx=5, pady=6)
    button9.configure(**button_style)
    button9.pack(padx=5, pady=7)
    close_button.pack(padx=5, pady=8)

    win.mainloop()

button = Button(root, text="Show Graphs", command=lambda: popup(), font="Raleway", bg="#20bebe",
fg="white", height=2,
                width=10)
button.configure(**button_style)
button.pack(padx=10, pady=10)
exit_button = Button(root, text="Exit", command=lambda: root.destroy(), height=1, width=4)
exit_button.pack(padx=10, pady=15)

root.mainloop()
```