Αναφορά Εργαστηριακής Άσκησης Python

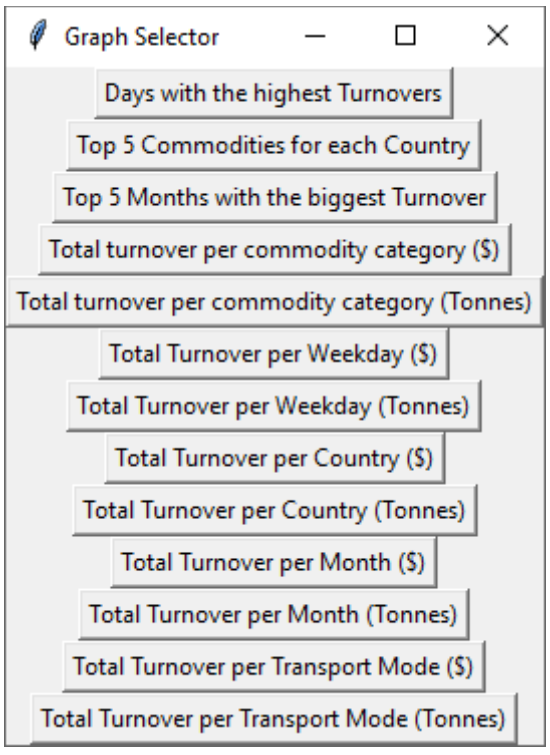
ΔΕΛΗΜΠΑΛΤΑΔΑΚΗΣ ΓΡΗΓΟΡΙΟΣ
ΑΜ: 1084647

Περιεχόμενα

Εξήγηση κώδικα

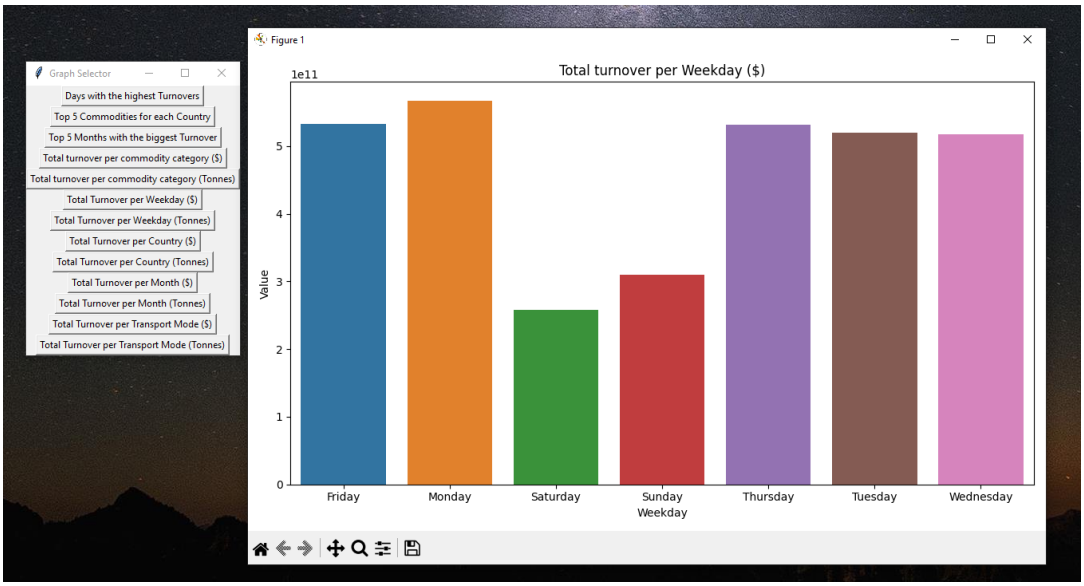
Θα παραθέσω μια σύντομη εξήγηση του κώδικα, που βρίσκεται στο τέλος της αναφοράς.

- Στην αρχή, εισάγονται απαραίτητες βιβλιοθήκες όπως **pandas** (για την δημιουργία data frames), **matplotlib**, **seaborn** (για τα γραφήματα), **os** (για την δημιουργία φακέλων) και **tkinter** (για την δημιουργία του γραφικού περιβάλλοντος), **sqlalchemy** (για την σύνδεση με την βάση δεδομένων)
- Στη συνέχεια, ορίζεται μια συνάρτηση **load_data()** που φορτώνει τα δεδομένα από το URL που μας δίνεται και δημιουργεί μια σύνδεση με την βάση δεδομένων MySQL.
- Ορίζουμε την **create_graph(),** μια συνάρτηση που δημιουργεί ένα γράφημα με βάση τα δεδομένα που παρέχονται, το αποθηκεύει και στην συνέχεια το εμφανίζει.
- Οι επόμενες συναρτήσεις (**monthly_turnover()**, **country_turnover()**, **transport_turnover()** κ.λπ.) υπολογίζουν τα ζητούμενα στατιστικά στοιχεία και τα οποία αποθηκεύει σε πίνακες SQL και αρχεία CSV. Επίσης σε αυτές τις συναρτήσεις καλείται η **create_graph()**.
- Στο κυρίως σκέλος του προγράμματος, δημιουργούνται δύο φάκελοι για τα παραγόμενα δεδομένα ('graphs' και 'csv_files') και φορτώνονται τα δεδομένα.
- Στη συνέχεια, δημιουργείται ένα απλό παράθυρο Tkinter με κάποια κουμπιά. Κάθε κουμπί καλεί μια διαφορετική συνάρτηση όταν πατηθεί, επιτρέποντας στον χρήστη να επιλέξει ποιο γράφημα θέλει να δημιουργήσει.

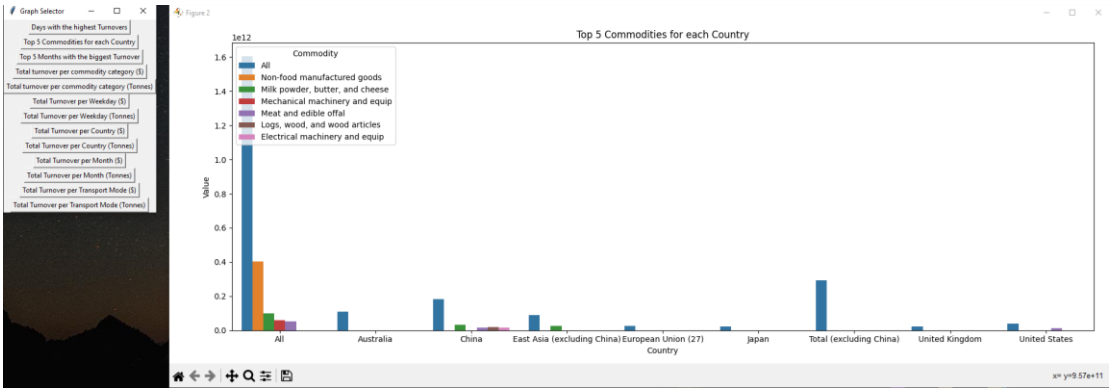


Screenshots παραδειγμάτων εφαρμογής

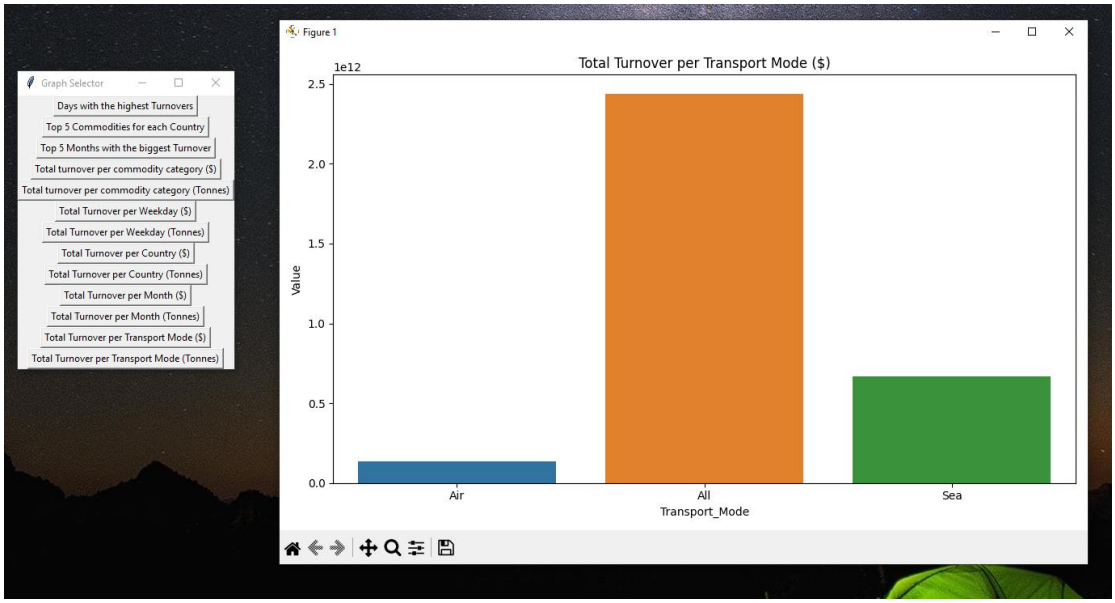
1. Κουμπί “Total Turnover per Week day ($)”:



2. Κουμπί “Top 5 Commodities for each Country”:

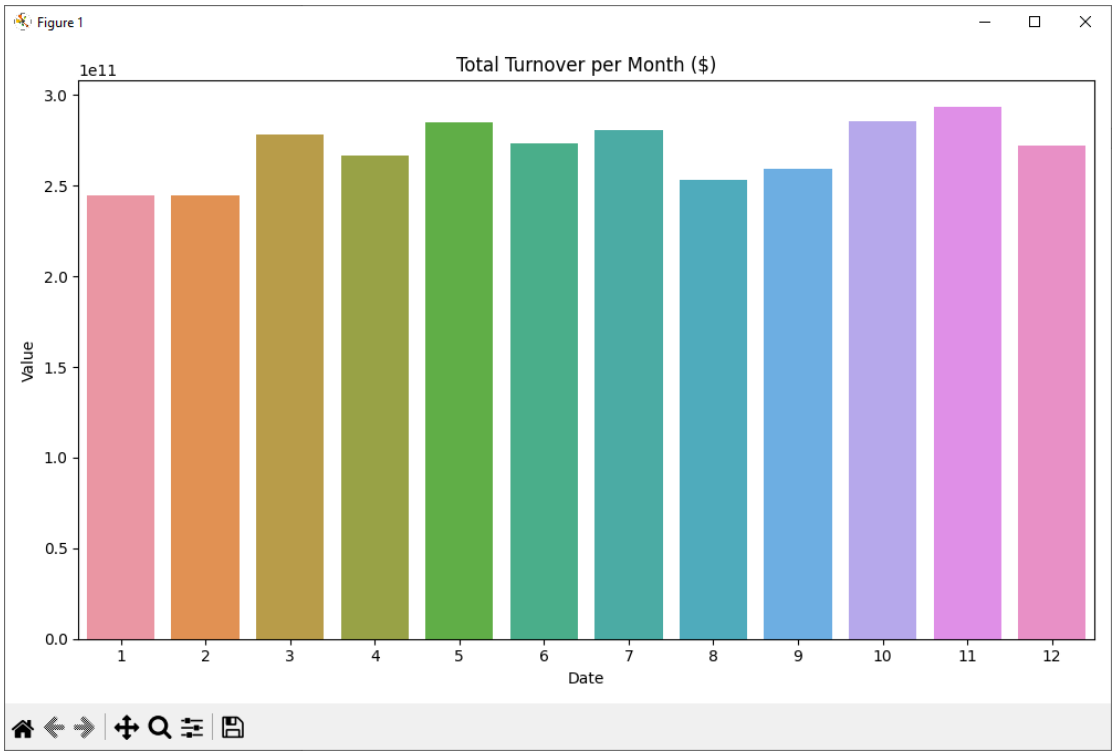


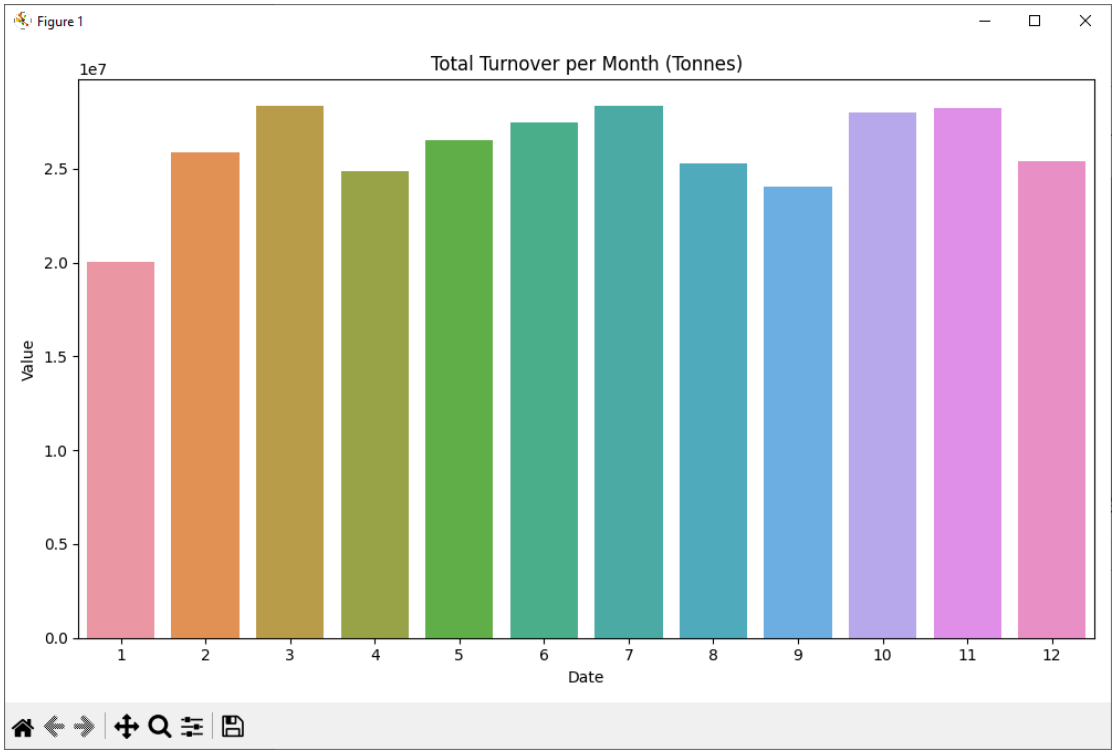
3. Κουμπί “Total Turnover per Transport Mode ($)”:



Τα ζητούμενα γραφήματα

1)





2)

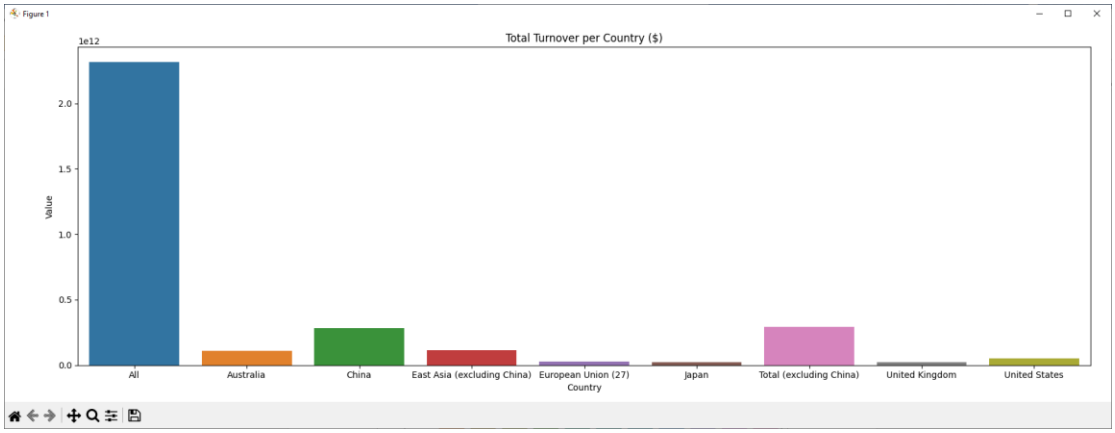


Figure 1
Total Turnover per Country ($)
1e12
2.0
1.5
1.0
0.5
0.0
Value
All
Australia
China
East Asia (excluding China)
European Union (27)
Japan
Total (excluding China)
United Kingdom
United States
Country

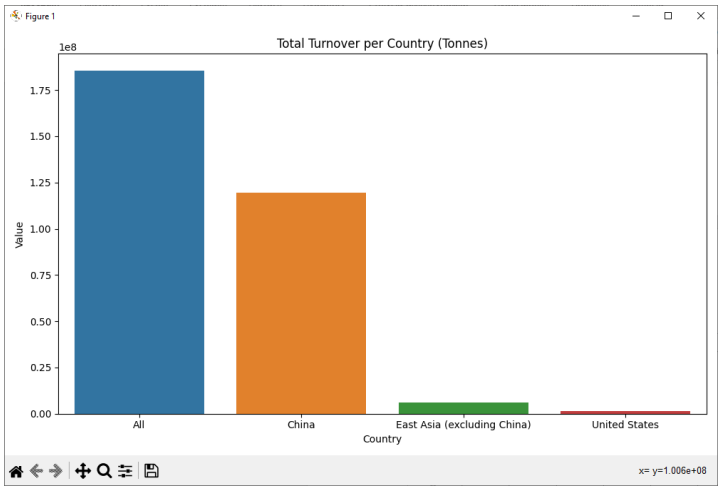


Figure 1
Total Turnover per Country (Tonnes)
1e8
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
Value
All
China
East Asia (excluding China)
United States
Country
x= y=1.006e+08

3)

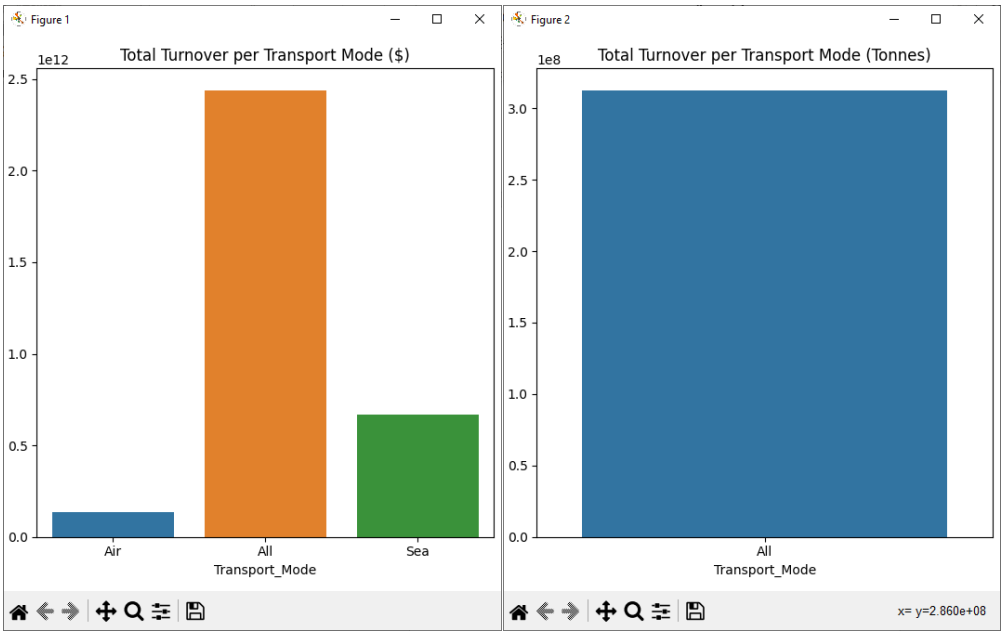


Figure 1
Total Turnover per Transport Mode ($)
1e12
2.5
2.0
1.5
1.0
0.5
0.0
Air
All
Sea
Transport_Mode
Figure 2
Total Turnover per Transport Mode (Tonnes)
1e8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
All
Transport_Mode
x= y=2.860e+08

4)

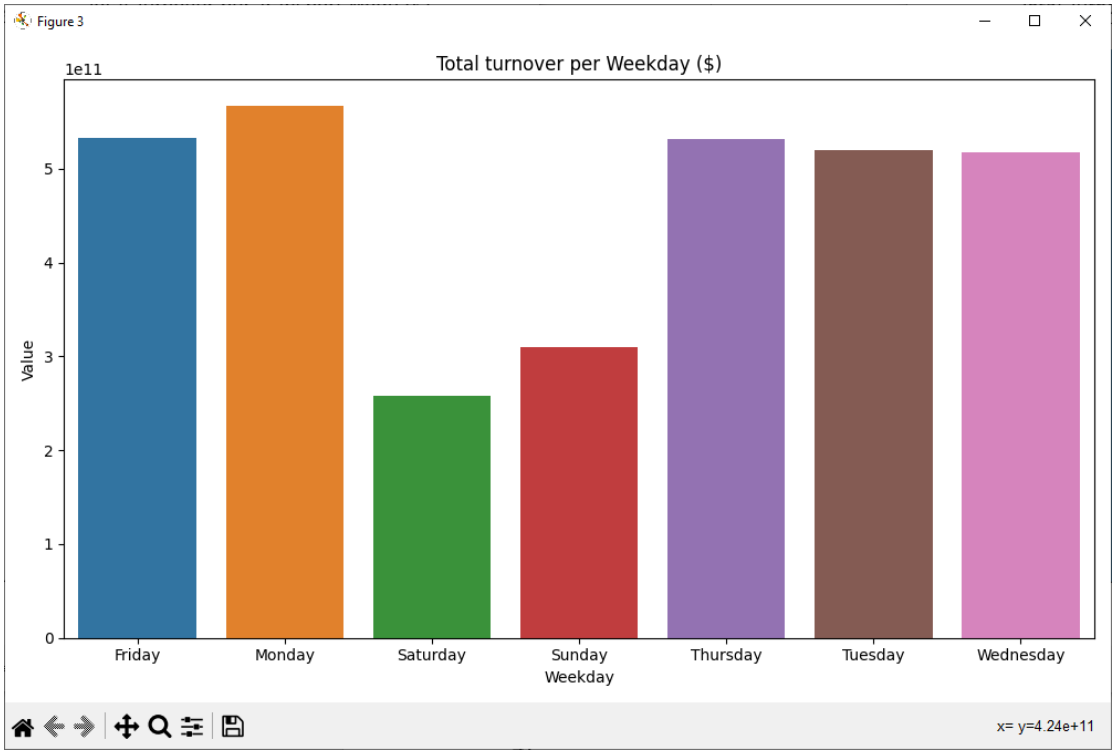


Figure 3
Total turnover per Weekday ($)
1e11
Value
0
1
2
3
4
5
Friday
Monday
Saturday
Sunday
Thursday
Tuesday
Wednesday
Weekday
x= y=4.24e+11

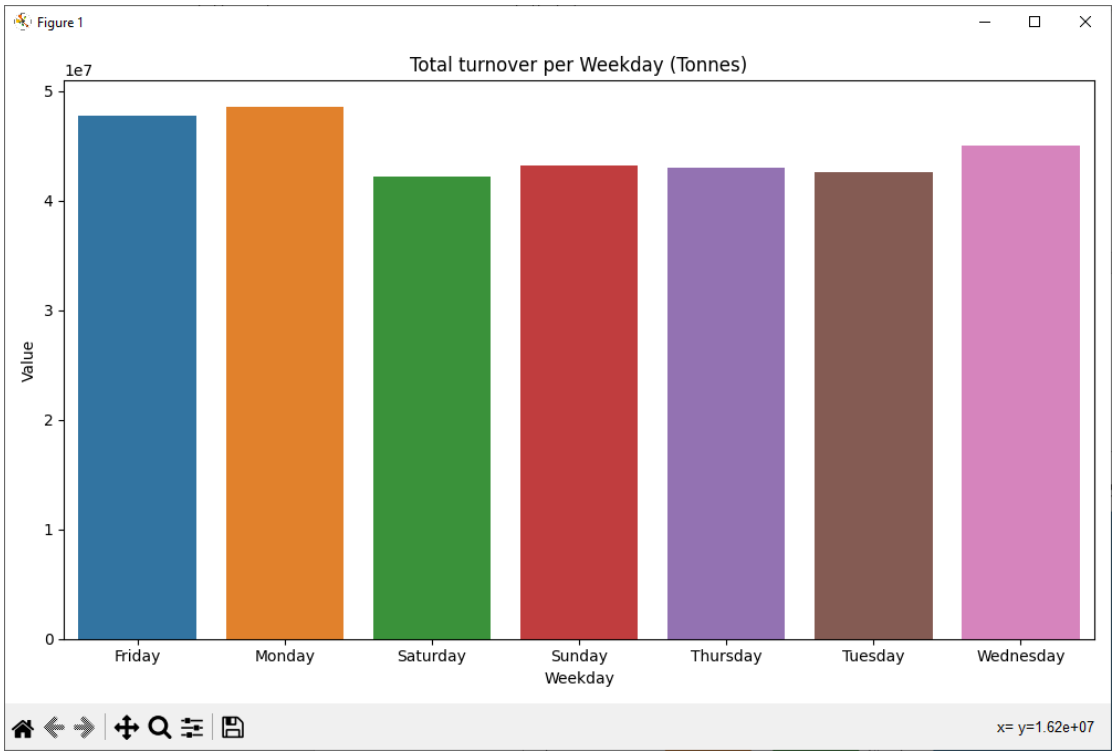


Figure 1
Total turnover per Weekday (Tonnes)
1e7
Value
0
1
2
3
4
5
Friday
Monday
Saturday
Sunday
Thursday
Tuesday
Wednesday
Weekday
x= y=1.62e+07

5)

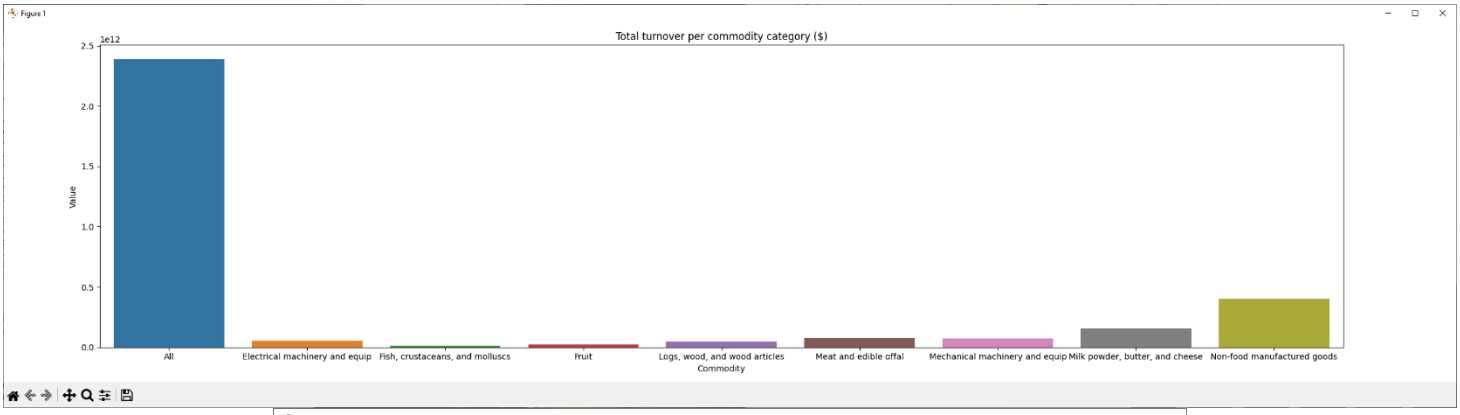


Figure 1
Total turnover per commodity category ($)
1e12
2.5
2.0
1.5
1.0
0.5
0.0
Value
All
Electrical machinery and equip
Fish, crustaceans, and molluscs
Fruit
Logs, wood, and wood articles
Meat and edible offal
Mechanical machinery and equip
Milk powder, butter, and cheese
Non-food manufactured goods
Commodity

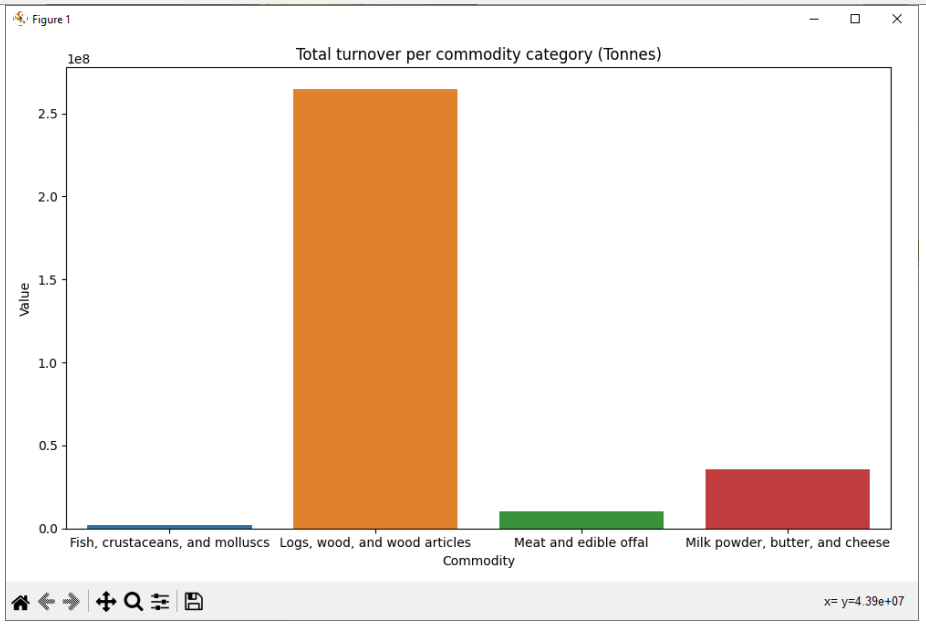


Figure 1
Total turnover per commodity category (Tonnes)
1e8
2.5
2.0
1.5
1.0
0.5
0.0
Value
Fish, crustaceans, and molluscs
Logs, wood, and wood articles
Meat and edible offal
Milk powder, butter, and cheese
Commodity
x= y=4.39e+07

6)

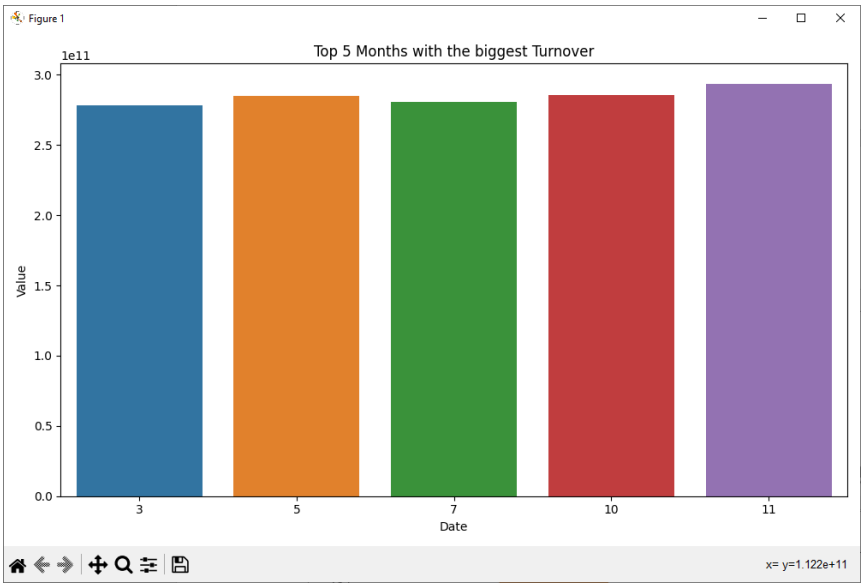


Figure 1
Top 5 Months with the biggest Turnover
1e11
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Value
3
5
7
10
11
Date
x= y=1.122e+11

7)

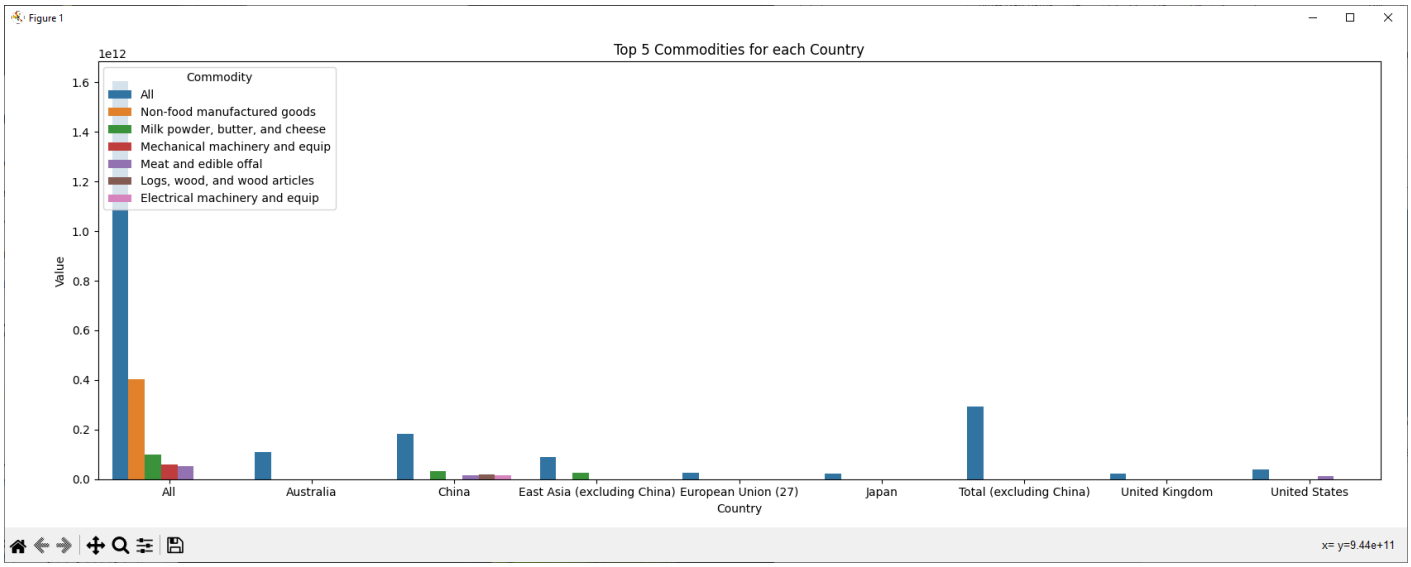


Figure 1
Top 5 Commodities for each Country
1e12
Commodity
All
Non-food manufactured goods
Milk powder, butter, and cheese
Mechanical machinery and equip
Meat and edible offal
Logs, wood, and wood articles
Electrical machinery and equip
Value
Country
All
Australia
China
East Asia (excluding China)
European Union (27)
Japan
Total (excluding China)
United Kingdom
United States
x= y=9.44e+11

8)

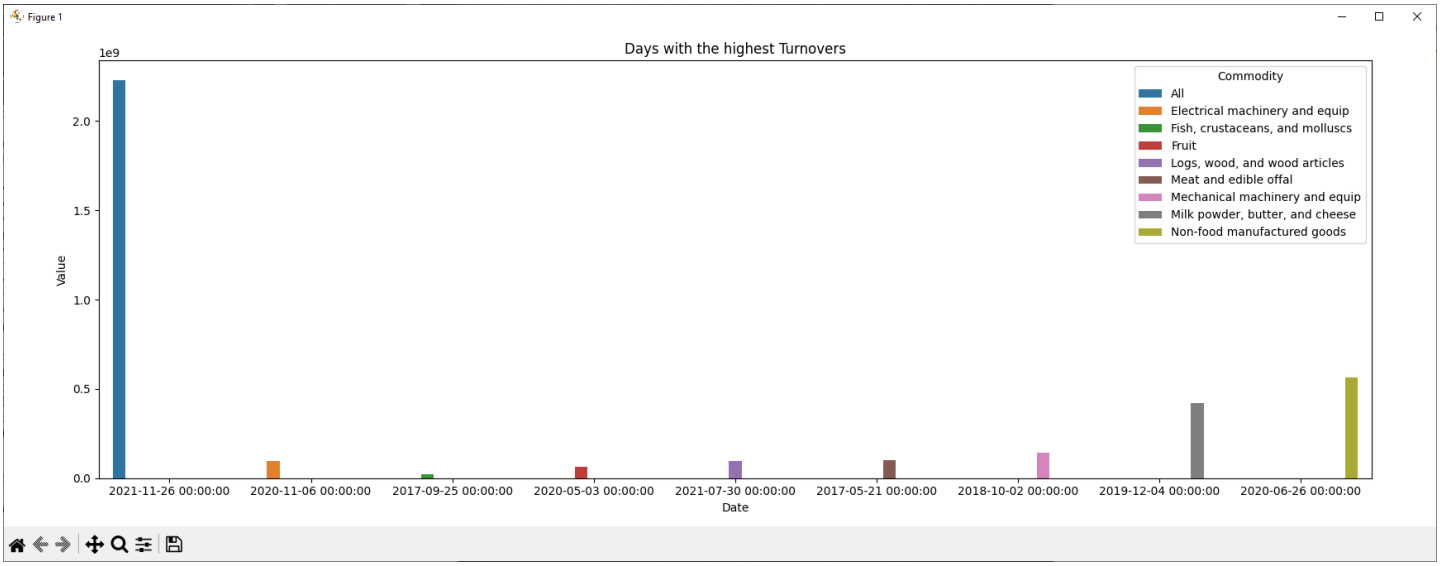
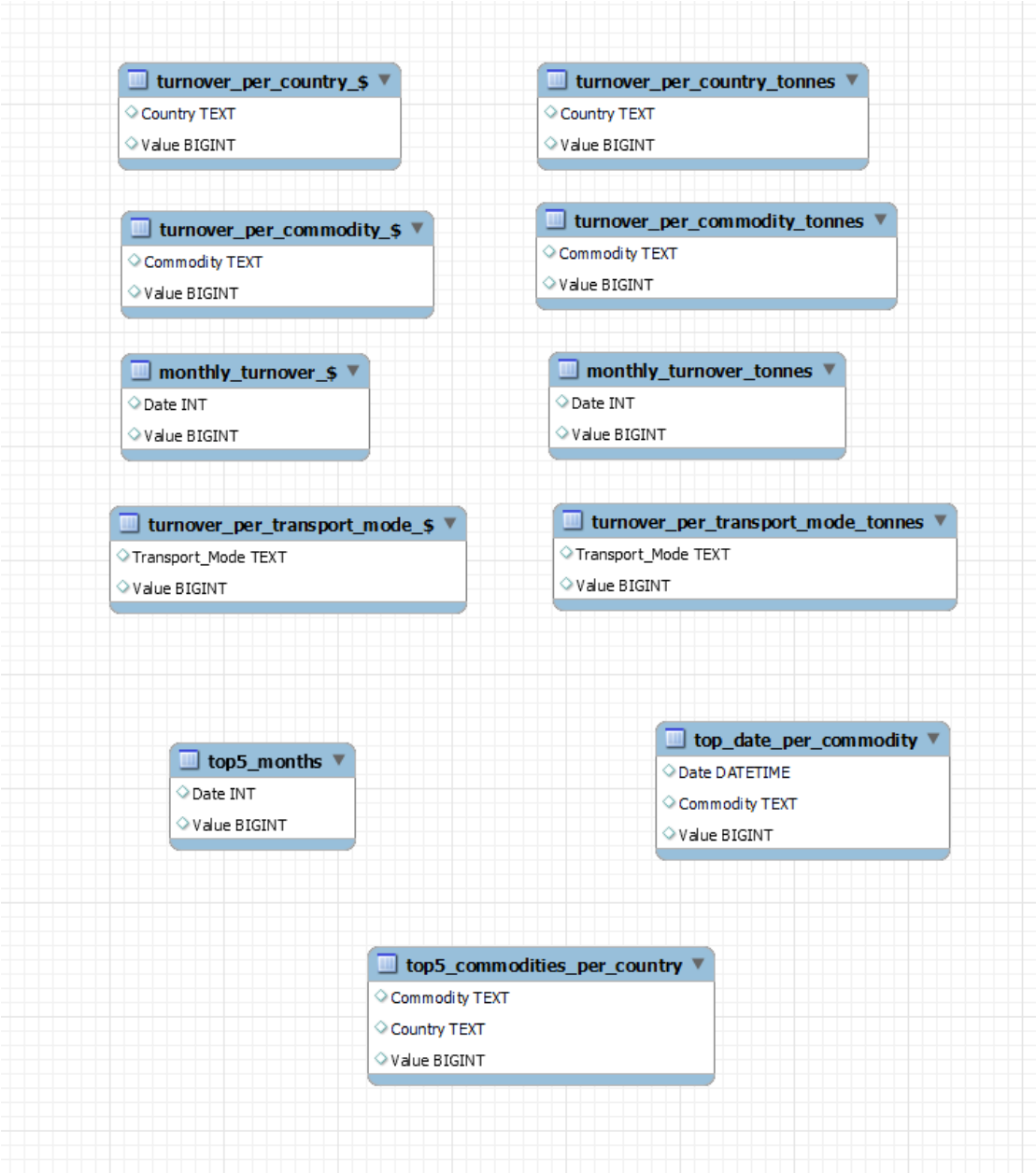


Figure 1
Days with the highest Turnovers
1e9
Commodity
All
Electrical machinery and equip
Fish, crustaceans, and molluscs
Fruit
Logs, wood, and wood articles
Meat and edible offal
Mechanical machinery and equip
Milk powder, butter, and cheese
Non-food manufactured goods
Value
Date
2021-11-26 00:00:00
2020-11-06 00:00:00
2017-09-25 00:00:00
2020-05-03 00:00:00
2021-07-30 00:00:00
2017-05-21 00:00:00
2018-10-02 00:00:00
2019-12-04 00:00:00
2020-06-26 00:00:00

Σχήμα της βάσης δεδομένων



Σχόλια – Παραδοχές

- Για να λειτουργήσει το πρόγραμμα θα πρέπει να έχει ήδη δημιουργηθεί μία βάση δεδομένων MySQL τοπικά και να έχετε κάνει pip install της βιβλιοθήκες που αναφέρονται στην εισαγωγή.
- Επίσης εάν θέλετε να τρέξετε τον κώδικα θα πρέπει να συνδεθείτε στην τοπική σας βάση ως εξής:
  Στο τέλος του αρχείου κώδικα **main.py** εισάγετε τα στοιχεία σας στην συνάρτηση **load_data(<username (string)>, <password (string)>, <dbase name (string)>)**
  πχ.

```
103
104  load_data('root', '12345678', 'covid_19_data')
105
```

```
mysql> use covid_19_data
Database changed
mysql> show tables;
+-----------------------------------+
| Tables_in_covid_19_data           |
+-----------------------------------+
| monthly_turnover_$                |
| monthly_turnover_tonnes           |
| top5_commodities_per_country      |
| top5_months                       |
| top_date_per_commodity            |
| turnover_per_commodity_$          |
| turnover_per_commodity_tonnes     |
| turnover_per_country_$            |
| turnover_per_country_tonnes       |
| turnover_per_transport_mode_$     |
| turnover_per_transport_mode_tonnes |
| turnover_per_week_$               |
| turnover_per_week_tonnes          |
+-----------------------------------+
13 rows in set (0.02 sec)
```

Εικόνα από την τοπική μου βάση μετά την εκτέλεση του προγράμματος

Κώδικας σε python

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sqlalchemy import create_engine
import os
import tkinter as tk

def load_data(username, password, dbase_name):
    # Load the data
    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"
    global data
    data = pd.read_csv(url) # import data into a pandas data frame
    # Convert Date to datetime
    data['Date'] = pd.to_datetime(data['Date'], format='%d/%m/%Y')

    # Create a connection to your MySQL server
    global engine
    engine = create_engine(f"mysql+pymysql://{username}:{password}@localhost:3306/{dbase_name}")

# Create the graphs
def create_graph(qdata, title, x_axis, y_axis, hue=None):
    # Create a figure and a set of subplots
    fig, ax = plt.subplots(figsize=(10, 6))
    hue = hue
    # Create a bar plot
    sns.barplot(data=qdata, x=x_axis, y=y_axis, hue=hue, ax=ax)
    plt.title(title)
    plt.tight_layout()
    plt.savefig(f"graphs/{title}.png")
    plt.show()


# Kathe mia apo tis parakatw synartiseis ftiaxnei enan apo tous zitoumenous pinakes,
# ton opoio kanei plot, apothikeuei stin mysql vasi kai kanei export ws csv

def monthly_turnover(measure):
    # Total turnover per month for each Measure
```

```
    monthly_turnover = data[data['Measure'] ==
measure].groupby(data['Date'].dt.month)['Value'].sum().reset_index()
    # Prosthiki stin vasi
    monthly_turnover.to_sql(name=f"monthly_turnover_{measure.lower()}",
con=engine, if_exists='replace', index=False)
    monthly_turnover.to_csv(f"csv_files/monthly_turnover_{measure.lower
()}.csv", index=False)
    create_graph(monthly_turnover, f"Total Turnover per Month
({measure})", 'Date', 'Value')

def country_turnover(measure):
    # Total turnover per country each Measure
    country_turnover = data[data['Measure'] ==
measure].groupby('Country')['Value'].sum().reset_index()
    country_turnover.to_sql(name=f"turnover_per_country_{measure.lower(
)}", con=engine, if_exists='replace', index=False)
    country_turnover.to_csv(f"csv_files/turnover_per_country_{measure.l
ower()}.csv", index=False)
    create_graph(country_turnover, f"Total Turnover per Country
({measure})", 'Country', 'Value')

def transport_turnover(measure):
    # Total turnover per transport mode for each Measure
    transport_turnover = data[data['Measure'] ==
measure].groupby('Transport_Mode')['Value'].sum().reset_index()
    transport_turnover.to_sql(name=f"turnover_per_transport_mode_{measu
re.lower()}", con=engine, if_exists='replace', index=False)
    transport_turnover.to_csv(f"csv_files/turnover_per_transport_mode_{
measure.lower()}.csv", index=False)
    create_graph(transport_turnover,f"Total Turnover per Transport Mode
({measure})", 'Transport_Mode', 'Value')

def weekday_turnover(measure):
    # Total turnover per weekday for each Measure
    weekday_turnover = data[data['Measure'] ==
measure].groupby('Weekday')['Value'].sum().reset_index()
    weekday_turnover.to_sql(name=f"turnover_per_week_{measure.lower()}"
, con=engine, if_exists='replace', index=False)
    weekday_turnover.to_csv(f"csv_files/turnover_per_week_{measure.lowe
r()}.csv", index=False)
    create_graph(weekday_turnover, f"Total turnover per Weekday
({measure})", 'Weekday', 'Value')

def commodity_turnover(measure):
    # Total turnover per commodity category for each Measure
    commodity_turnover = data[data['Measure'] ==
measure].groupby('Commodity')['Value'].sum().reset_index()
```

```
    commodity_turnover.to_sql(name=f"turnover_per_commodity_{measure.lo
wer()}", con=engine, if_exists='replace', index=False)
    commodity_turnover.to_csv(f"csv_files/turnover_per_commodity_{measu
re.lower()}.csv", index=False)
    create_graph(commodity_turnover, f"Total turnover per commodity
category ({measure})", 'Commodity', 'Value')

def top5_months():
    # Top 5 Months with the biggest Turnover
    top5_months =
data.groupby(data['Date'].dt.month)['Value'].sum().nlargest(5).reset_in
dex()
    top5_months.to_sql(name=f"top5_months", con=engine,
if_exists='replace', index=False)
    top5_months.to_csv(f"csv_files/top5_months.csv", index=False)
    create_graph(top5_months, 'Top 5 Months with the biggest Turnover',
'Date', 'Value')

def top5_commodities_per_country():
    # Top 5 Commodities with the biggest Turnover for each Country
    commodity_country_turnover = data.groupby(['Commodity',
'Country'])['Value'].sum().reset_index()
    # Sort by 'Country' and 'Value', then group by 'Country' and take
the top 5 commodities for each country
    top5_commodities_per_country =
commodity_country_turnover.sort_values(['Country', 'Value'],
ascending=[True, False]).groupby('Country').head(5)
    top5_commodities_per_country.to_sql(name=f"top5_commodities_per_cou
ntry", con=engine, if_exists='replace', index=False)
    top5_commodities_per_country.to_csv(f"csv_files/top5_commodities_pe
r_country.csv", index=False)
    create_graph(top5_commodities_per_country, 'Top 5 Commodities for
each Country', 'Country', 'Value', 'Commodity')

def top_date_per_commodity():
    # Calculate total turnover for each date for each commodity
    date_commodity_turnover = data.groupby(['Date',
'Commodity'])['Value'].sum().reset_index()
    # Sort by 'Commodity' and 'Value', then group by 'Commodity' and
take the date with the highest turnover for each commodity
    top_date_per_commodity =
date_commodity_turnover.sort_values(['Commodity', 'Value'],
ascending=[True, False]).groupby('Commodity').head(1)
    top_date_per_commodity.to_sql(name=f"top_date_per_commodity",
con=engine, if_exists='replace', index=False)
    top_date_per_commodity.to_csv(f"csv_files/top_date_per_commodity.cs
v", index=False)
```

```
    create_graph(top_date_per_commodity, 'Days with the highest
Turnovers', 'Date', 'Value', 'Commodity')

#main:

# Dhmiourgia fakelou me tis eikones grafimatwn
os.makedirs('graphs', exist_ok=True)
# Dhmiourgia fakelou me ta csv
os.makedirs('csv_files', exist_ok=True)

load_data('root', 'password', 'dbase-name')

# GUI Menu

# Create a new Tkinter window
window = tk.Tk()
window.title("Graph Selector")

# Create buttons
button1 = tk.Button(window, text="Days with the highest Turnovers",
command=lambda: top_date_per_commodity())
button2 = tk.Button(window, text="Top 5 Commodities for each Country",
command=lambda: top5_commodities_per_country())
button3 = tk.Button(window, text="Top 5 Months with the biggest
Turnover", command=lambda: top5_months())
button4 = tk.Button(window, text="Total turnover per commodity category
($)", command=lambda: commodity_turnover('$'))
button5 = tk.Button(window, text="Total turnover per commodity category
(Tonnes)", command=lambda: commodity_turnover('Tonnes'))
button6 = tk.Button(window, text="Total Turnover per Weekday ($)",
command=lambda: weekday_turnover('$'))
button7 = tk.Button(window, text="Total Turnover per Weekday (Tonnes)",
command=lambda: weekday_turnover('Tonnes'))
button8 = tk.Button(window, text="Total Turnover per Country ($)",
command=lambda: country_turnover('$'))
button9 = tk.Button(window, text="Total Turnover per Country (Tonnes)",
command=lambda: country_turnover('Tonnes'))
button10 = tk.Button(window, text="Total Turnover per Month ($)",
command=lambda: monthly_turnover('$'))
button11 = tk.Button(window, text="Total Turnover per Month (Tonnes)",
command=lambda: monthly_turnover('Tonnes'))
button12 = tk.Button(window, text="Total Turnover per Transport Mode
($)", command=lambda: transport_turnover('$'))
button13 = tk.Button(window, text="Total Turnover per Transport Mode
(Tonnes)", command=lambda: transport_turnover('Tonnes'))

# Pack the buttons so they are visible
button1.pack()
```

```
button2.pack()
button3.pack()
button4.pack()
button5.pack()
button6.pack()
button7.pack()
button8.pack()
button9.pack()
button10.pack()
button11.pack()
button12.pack()
button13.pack()

# Start the event loop
window.mainloop()
```