

# Python Project 2023



## Στοιχεία Φοιτητή

---

Όνομα: Ιάσοντας

Επώνυμο: Παυλόπουλος

ΑΜ: 1084565

Έτος: 3<sup>ο</sup>

Email: up1084565@upnet.gr



**CEID @ UNIVERSITY of PATRAS**  
Computer Engineering & Informatics Department

# Περιεχόμενα

---

1. **Εισαγωγή**
2. **Κώδικας Python**
3. **Screenshots Παραδειγμάτων της εφαρμογής και σχήμα Βάσης Δεδομένων**
4. **Ζητούμενα Γραφήματα**
5. **Σχόλια-Παραδοχές Υλοποίησης**
6. **Βιβλιογραφία-Παραπομπές**

# Εισαγωγή

---

Πριν ξεκινήσω την παρουσίαση του Project της Python, θα ήθελα να σας ενημερώσω πως όλα τα resource files, όπως το Script της Python (script.py), τα .csv αρχεία, το αρχείο SQL, το σχήμα της βάσης, συμπεριλαμβανομένης και αυτής της αναφοράς μπορείτε να τα βρείτε στο παρακάτω Github Repository. Επίσης μπορείτε να δείτε τις αλλαγές που έχουν γίνει από την αρχή δημιουργίας του Project (commits).

**Github Repository:** <https://github.com/CallMeJasonYT/Python-Project-2023>

Για το κατέβασμα του αρχείου των δεδομένων και τον έλεγχο ύπαρξης του, χρησιμοποιήθηκαν οι βιβλιοθήκες **urllib** και **os** αντίστοιχα.

Για την σύνδεση με την βάση δεδομένων χρησιμοποιήθηκε η βιβλιοθήκη **sqlalchemy**

Για τα διαγράμματα χρησιμοποιήθηκαν οι βιβλιοθήκες **pandas** και **matplotlib**.

Για την υλοποίηση του GUI χρησιμοποιήθηκε η βιβλιοθήκη **customtkinter**, η οποία είναι μία custom βιβλιοθήκη σαν επέκταση της **tkinter**.

*Περισσότερες πληροφορίες για την εγκατάστασή των βιβλιοθηκών και την προέλευσή τους ανατρέξτε στο Παράρτημα 5 και 6*

# Κώδικας Python

---

```
import customtkinter
import pandas as pd
import matplotlib.pyplot as plt
import urllib.request
import os.path
from sqlalchemy import create_engine

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"
filename = "data.csv"

if not os.path.isfile(filename):
    urllib.request.urlretrieve(url, filename)
else:
    print("File already exists, skipping download.")

# ----- GUI Code -----#
customtkinter.set_appearance_mode("System")
customtkinter.set_default_color_theme("blue")

engine = create_engine(
    "mysql://root:root@localhost/covid_effects"
) # Engine Creation for SQL Connection
engine.dispose()
```

```

class App(customtkinter.CTk):

    def __init__(self):
        super().__init__()

        # Configure Main window, center it and resize it
        self.title("Project Python 2023.py")
        self.geometry(f"{1280}x{720}")
        screen_width = self.winfo_screenwidth()
        screen_height = self.winfo_screenheight()
        x = (screen_width - 1280) // 2
        y = (screen_height - 720) // 2
        self.geometry("+{}+{}".format(x, y))

        # Configure grid layout
        self.grid_columnconfigure(1, weight=1)
        self.grid_columnconfigure((2, 3), weight=0)
        self.grid_rowconfigure((0, 1, 2), weight=1)

        # Create sidebar frame with Buttons
        self.sidebar_frame = customtkinter.CTkFrame(self, width=140, corner_radius=0)
        self.sidebar_frame.grid(row=0, column=0, rowspan=4, sticky="nsew")
        self.sidebar_frame.grid_rowconfigure(9, weight=1)
        self.logo_label = customtkinter.CTkLabel(
            self.sidebar_frame,
            text="Project Python",
            font=customtkinter.CTkFont(size=20, weight="bold"),

```

```

)
self.logo_label.grid(row=0, column=0, padx=20, pady=(20, 10))
self.sidebar_button_1 = customtkinter.CTkButton(
    self.sidebar_frame,
    text="Profit per Month",
    height=(40),
    width=(300),
    font=customtkinter.CTkFont(size=18, weight="bold"),
    command=self.first_figure,
)
self.sidebar_button_1.grid(row=1, column=0, padx=20, pady=10)
self.sidebar_button_2 = customtkinter.CTkButton(
    self.sidebar_frame,
    text="Profit per Country",
    height=(40),
    width=(300),
    font=customtkinter.CTkFont(size=18, weight="bold"),
    command=self.second_figure,
)
self.sidebar_button_2.grid(row=2, column=0, padx=20, pady=10)
self.sidebar_button_3 = customtkinter.CTkButton(
    self.sidebar_frame,
    text="Profit per Transport",
    height=(40),
    width=(300),
    font=customtkinter.CTkFont(size=18, weight="bold"),
    command=self.third_figure,

```

```

)
self.sidebar_button_3.grid(row=3, column=0, padx=20, pady=10)
self.sidebar_button_4 = customtkinter.CTkButton(
    self.sidebar_frame,
    text="Profit per Day of Week",
    height=(40),
    width=(300),
    font=customtkinter.CTkFont(size=18, weight="bold"),
    command=self.fourth_figure,
)
self.sidebar_button_4.grid(row=4, column=0, padx=20, pady=10)
self.sidebar_button_5 = customtkinter.CTkButton(
    self.sidebar_frame,
    text="Profit per Product",
    height=(40),
    width=(300),
    font=customtkinter.CTkFont(size=18, weight="bold"),
    command=self.fifth_figure,
)
self.sidebar_button_5.grid(row=5, column=0, padx=20, pady=10)
self.sidebar_button_6 = customtkinter.CTkButton(
    self.sidebar_frame,
    text="5 Most Profitable Months",
    height=(40),
    width=(300),
    font=customtkinter.CTkFont(size=18, weight="bold"),
    command=self.sixth_figure,

```

```

)
self.sidebar_button_6.grid(row=6, column=0, padx=20, pady=10)
self.sidebar_button_7 = customtkinter.CTkButton(
    self.sidebar_frame,
    text="5 Most Profitable Products",
    height=(40),
    width=(300),
    font=customtkinter.CTkFont(size=18, weight="bold"),
    command=self.seventh_figure,
)
self.sidebar_button_7.grid(row=7, column=0, padx=20, pady=10)
self.sidebar_button_8 = customtkinter.CTkButton(
    self.sidebar_frame,
    text="Most Profitable Day per Product",
    height=(40),
    width=(300),
    font=customtkinter.CTkFont(size=18, weight="bold"),
    command=self.eighth_figure,
)
self.sidebar_button_8.grid(row=8, column=0, padx=20, pady=10)
self.sidebar_button_9 = customtkinter.CTkButton(
    self.sidebar_frame,
    text="Exit",
    height=(40),
    width=(300),
    font=customtkinter.CTkFont(size=18, weight="bold"),
    command=lambda: [self.destroy(), plt.close("all")],

```



```

        fg_color="Red",
    )

    self.sidebar_button_9.grid(row=9, column=0, padx=20, pady=10)

# ----- Graphs Functions Code -----#

def first_figure(self):
    plt.close("all")

    usecols = [
        "Direction",
        "Date",
        "Measure",
        "Value",
        "Year",
    ] # Define the Used Columns for this Graph

    data = pd.read_csv(
        "data.csv", usecols=usecols
    ) # Read the .csv file and save it in a DataFrame

    data = data[
        data["Direction"] == "Exports"
    ] # Accept only the 'Exports' Direction

    data["Month"] = pd.to_datetime(data["Date"], format="%d/%m/%Y").dt.strftime(
        "%b"
    )

    ) # Create a new column for short month names

    monthly_data = (
        data.groupby(["Year", "Month", "Measure"])["Value"].sum().reset_index()

    ) # Group the data by Year, Month and Measure and sum the Value column for $ and Tonnes
    measures separately

```

```

month_order = [
    "Jan",
    "Feb",
    "Mar",
    "Apr",
    "May",
    "Jun",
    "Jul",
    "Aug",
    "Sep",
    "Oct",
    "Nov",
    "Dec",
] # Months Order

monthly_data["Month"] = pd.Categorical(
    monthly_data["Month"], categories=month_order, ordered=True
) # Ordering the Columns By Months

```

# Create pivot tables with years as columns, months as rows, and the sum of values as values for \$ and Tonnes measures separately

```

monthly_dollars = monthly_data[monthly_data["Measure"] == "$"].pivot_table(
    index="Month", columns="Year", values="Value", aggfunc="sum"
)

monthly_tonnes = monthly_data[monthly_data["Measure"] == "Tonnes"].pivot_table(
    index="Month", columns="Year", values="Value", aggfunc="sum"
)

```

```

# Passing the data into the SQL Database
monthly_data[monthly_data["Measure"] == "$"].to_sql(
    "monthly_profit", con=engine, if_exists="replace", index=False
)
monthly_data[monthly_data["Measure"] == "Tonnes"].to_sql(
    "monthly_profit", con=engine, if_exists="append", index=False
)

# Exporting the data into .csv files
monthly_dollars.to_csv("monthly_dollars.csv")
monthly_tonnes.to_csv("monthly_tonnes.csv")

fig, axs = plt.subplots(
    num="Profit per Month", nrow=2, figsize=(9.39, 6.48)
) # Create the bar charts as subplots

# Plot the graphs
monthly_dollars.plot(kind="bar", ax=axs[0], legend=False)
axs[0].set_title("Monthly Value by Year ($)")
axs[0].set_xlabel("Month")
axs[0].set_ylabel("Value ($)")

fig.legend(loc="upper right", ncol=len(month_order))
monthly_tonnes.plot(kind="bar", ax=axs[1], legend=False)
axs[1].set_title("Monthly Value by Year (Tonnes)")
axs[1].set_xlabel("Month")
axs[1].set_ylabel("Value (Tonnes)")

```

```

fig.canvas.manager.set_window_title("Profit per Month")
mngr = plt.get_current_fig_manager()
mngr.window.geometry("+661+210")
plt.tight_layout(pad=3.0)
plt.show()

```

```

def second_figure(self):
    plt.close("all")
    usecols = [
        "Direction",
        "Measure",
        "Value",
        "Country",
    ] # Define the Used Columns for this Graph
    data = pd.read_csv(
        "data.csv", usecols=usecols
    ) # Read the .csv file and save it
    data = data[
        data["Direction"] == "Exports"
    ] # Accept only the 'Exports' Direction
    data = (
        data.groupby(["Country", "Measure"])["Value"].sum().reset_index()
    ) # Group the Value by Country and Measure and sum for $ and Tonnes measures separately

    # Make the data names Shorter
    data = data.replace(

```

```

[
    "East Asia (excluding China)",
    "European Union (27)",
    "Total (excluding China)",
    "United Kingdom",
    "United States",
],
["EAsia-China", "EU (27)", "Total-China", "UK", "USA"],
)

# Diverse the data based on the Measure Value
dollars_country = data[data["Measure"] == "$"]
tonnes_country = data[data["Measure"] == "Tonnes"]

# Passing the data into the SQL Database
dollars_country.to_sql(
    "country_profit", con=engine, if_exists="replace", index=False
)
tonnes_country.to_sql(
    "country_profit", con=engine, if_exists="append", index=False
)

# Exporting the data into .csv files
dollars_country.to_csv("dollars_country.csv")
tonnes_country.to_csv("tonnes_country.csv")

fig, axs = plt.subplots(

```

```

num="Profit per Country", nrow=2, figsize=(9.39, 6.48)
) # Create the bar charts as subplots

# Plot the graphs
dollars_country.plot(
    x="Country", y="Value", kind="bar", ax=axes[0], legend=False
)
axes[0].set_title("Summary Value by Country ($)")
axes[0].set_xlabel("Country")
axes[0].set_ylabel("Value ($)")
tonnes_country.plot(
    x="Country", y="Value", kind="bar", ax=axes[1], legend=False
)
axes[1].set_title("Summary Value by Country (Tonnes)")
axes[1].set_xlabel("Country")
axes[1].set_ylabel("Value (Tonnes)")

fig.suptitle("Profit Per Country", fontweight="bold", fontsize=16)
mgr = plt.get_current_fig_manager()
mgr.window.geometry("+661+210")
plt.tight_layout()
plt.show()

def third_figure(self):
    plt.close("all")
    usecols = [
        "Direction",

```

```

    "Measure",
    "Value",
    "Transport_Mode",
] # Define the Used Columns for this Graph

data = pd.read_csv(
    "data.csv", usecols=usecols
) # Read the .csv file and save it

data = data[
    data["Direction"] == "Exports"
] # Accept only the 'Exports' Direction

data = (
    data.groupby(["Transport_Mode", "Measure"])["Value"].sum().reset_index()
) # Group the data by Transport_Mode and Measure and sum the Value column for $ and Tonnes
measures separately

# Diverse the data based on the Measure Value
dollars_transport = data[data["Measure"] == "$"]
tonnes_transport = data[data["Measure"] == "Tonnes"]

# Passing the data into the SQL Database
dollars_transport.to_sql(
    "transport_profit", con=engine, if_exists="replace", index=False
)
tonnes_transport.to_sql(
    "transport_profit", con=engine, if_exists="append", index=False
)

# Exporting the data into .csv files

```

```

dollars_transport.to_csv("dollars_transport.csv")
tonnes_transport.to_csv("tonnes_transport.csv")

fig, axs = plt.subplots(
    num="Profit per Transport", nrows=2, figsize=(9.39, 6.48)
) # Create the bar charts as subplots
# Plot the graphs
dollars_transport.plot(
    x="Transport_Mode", y="Value", kind="bar", ax=axs[0], legend=False
)
axs[0].set_title("Summary Value by Transport Mode ($)")
axs[0].set_xlabel("Transport")
axs[0].set_ylabel("Value ($)")
tonnes_transport.plot(
    x="Transport_Mode", y="Value", kind="bar", ax=axs[1], legend=False
)
axs[1].set_title("Summary Value by Transport Mode (Tonnes)")
axs[1].set_xlabel("Transport")
axs[1].set_ylabel("Value (Tonnes)")

fig.canvas.manager.set_window_title("Profit per Transport")
mngr = plt.get_current_fig_manager()
mngr.window.geometry("+661+210")
plt.tight_layout()
plt.show()

```

```

def fourth_figure(self):

```



```

plt.close("all")

usecols = [
    "Direction",
    "Date",
    "Measure",
    "Value",
] # Define the Used Columns for this Graph

data = pd.read_csv(
    "data.csv", usecols=usecols
) # Read the .csv file and save it

data = data[
    data["Direction"] == "Exports"
] # Accept only the 'Exports' Direction

data["Day"] = pd.to_datetime(data["Date"], format="%d/%m/%Y").dt.strftime(
    "%a"
) # Create a new column for short day names

daily_data = (
    data.groupby(["Day", "Measure"])["Value"].sum().reset_index()
) # Group the data by Day and Measure and sum the Value column for $ and Tonnes measures
separately

day_order = ["Mon", "Tue", "Wed", "Thu", "Fri", "Sat", "Sun"] # Days Order

daily_data["Day"] = pd.Categorical(
    daily_data["Day"], categories=day_order, ordered=True
) # Ordering the Columns By Days

daily_data = daily_data.sort_values("Day") # Sorting the Columns by Day

```

```

# Creating separate tables for the $ Measure and the Tonnes
dollars_per_day = daily_data[daily_data["Measure"] == "$"]
tonnes_per_day = daily_data[daily_data["Measure"] == "Tonnes"]

# Passing the data into the SQL Database
dollars_per_day.to_sql(
    "profit_per_day", con=engine, if_exists="replace", index=False
)
tonnes_per_day.to_sql(
    "profit_per_day", con=engine, if_exists="append", index=False
)

# Exporting the data into .csv files
dollars_per_day.to_csv("dollars_per_day.csv")
tonnes_per_day.to_csv("tonnes_per_day.csv")

fig, axs = plt.subplots(
    num="Profit per Day of Week", nrows=2, figsize=(9.39, 6.48)
) # Create the bar charts as subplots

# Plot the graphs
dollars_per_day.plot(x="Day", y="Value", kind="bar", ax=axs[0], legend=False)
axs[0].set_title("Total Daily Value ($)")
axs[0].set_xlabel("Day")
axs[0].set_ylabel("Value ($)")

tonnes_per_day.plot(x="Day", y="Value", kind="bar", ax=axs[1], legend=False)

```

```
axs[1].set_title("Total Daily Value (Tonnes)")
axs[1].set_xlabel("Day")
axs[1].set_ylabel("Value (Tonnes)")
```

```
mngr = plt.get_current_fig_manager()
mngr.window.geometry("+661+210")
plt.tight_layout()
plt.show()
```

```
def fifth_figure(self):
    plt.close("all")
    usecols = [
        "Direction",
        "Commodity",
        "Measure",
        "Value",
    ] # Define the Used Columns for this Graph
    data = pd.read_csv(
        "data.csv", usecols=usecols
    ) # Read the .csv file and save it
    data = data[
        data["Direction"] == "Exports"
    ] # Accept only the 'Exports' Direction
    data = (
        data.groupby(["Commodity", "Measure"])["Value"].sum().reset_index()
    ) # Group the data by Commodity and Measure and sum the Value column for $ and Tonnes
    measures separately
```

```
# Replace the Long names of the Commodity Values
```

```
data = data.replace(  
    [  
        "Milk powder, butter, and cheese",  
        "Meat and edible offal",  
        "Logs, wood, and wood articles",  
        "Fish, crustaceans, and molluscs",  
        "Non-food manufactured goods",  
        "Mechanical machinery and equip",  
        "Electrical machinery and equip",  
    ],  
    [  
        "Dairy",  
        "Meat",  
        "Wood",  
        "Fish",  
        "Other Goods",  
        "Mech Machines",  
        "E-Machines",  
    ],  
)
```

```
# Diverse the data based on the Measure Value
```

```
dollars_commodity = data[data["Measure"] == "$"]
```

```
tonnes_commodity = data[data["Measure"] == "Tonnes"]
```

```
# Passing the data into the SQL Database
```

```

dollars_commodity.to_sql(
    "commodity_profit", con=engine, if_exists="replace", index=False
)
tonnes_commodity.to_sql(
    "commodity_profit", con=engine, if_exists="append", index=False
)

# Exporting the data into .csv files
dollars_commodity.to_csv("dollars_commodity.csv")
tonnes_commodity.to_csv("tonnes_commodity.csv")

fig, axs = plt.subplots(
    num="Profit per Product", nrows=2, figsize=(9.39, 6.48)
) # Create the bar charts as subplots

# Plot the graphs
dollars_commodity.plot(
    x="Commodity", y="Value", kind="bar", ax=axs[0], legend=False
)
axs[0].set_title("Total Commodity Value ($)")
axs[0].set_xlabel("Commodity")
axs[0].set_ylabel("Value ($)")

tonnes_commodity.plot(
    x="Commodity", y="Value", kind="bar", ax=axs[1], legend=False
)
axs[1].set_title("Total Commodity Value (Tonnes)")

```

```

axs[1].set_xlabel("Commodity")
axs[1].set_ylabel("Value (Tonnes)")

mngr = plt.get_current_fig_manager()
mngr.window.geometry("+661+210")
plt.tight_layout()
plt.show()

```

```

def sixth_figure(self):
    plt.close("all")
    usecols = [
        "Direction",
        "Date",
        "Measure",
        "Value",
    ] # Define the Used Columns for this Graph
    data = pd.read_csv(
        "data.csv", usecols=usecols
    ) # Read the .csv file and save it

    data = data[
        (data["Direction"] == "Exports") & (data["Measure"] == "$")
    ] # Accept only the 'Exports' Direction and $
    data["Date"] = pd.to_datetime(data["Date"], format="%d/%m/%Y")

    data = (
        data.groupby(pd.Grouper(key="Date", freq="M"))["Value"].sum().reset_index()
    )

```

```

)

top_5_months = data.groupby(pd.Grouper(key="Date", freq="M"))["Value"].sum()
data = data.sort_values(by="Value", ascending=False).head(5)
top_5_months = top_5_months.sort_values(ascending=False).head(5)

# Passing the data into the SQL Database
data.to_sql("top5_months", con=engine, if_exists="replace", index=False)

# Exporting the data into .csv files
top_5_months.to_csv("top_5_months.csv")

plt.figure(
    figsize=(9.39, 6.48), num="5 Most Profitable Months"
) # Create the bar charts as subplots
plt.bar(top_5_months.index.strftime("%b %Y"), top_5_months.values)
plt.xlabel("Month", fontsize=14, labelpad=10)
plt.ylabel("Export Value", fontsize=14, labelpad=20)
plt.title(
    "Top 5 Most Profitable Months",
    fontweight="bold",
    fontsize=16,
    y=1.05,
)

mgr = plt.get_current_fig_manager()
mgr.window.geometry("+661+210")
plt.tight_layout()

```

```
plt.show()
```

```
def seventh_figure(self):
```

```
    plt.close("all")
```

```
    usecols = [
```

```
        "Direction",
```

```
        "Country",
```

```
        "Commodity",
```

```
        "Measure",
```

```
        "Value",
```

```
    ] # Define the Used Columns for this Graph
```

```
    data = pd.read_csv(
```

```
        "data.csv", usecols=usecols
```

```
    ) # Read the .csv file and save it
```

```
    data = data[(data["Direction"] == "Exports") & (data["Measure"] == "$")]
```

```
    # Make the data names Shorter
```

```
    data = data.replace(
```

```
        [
```

```
            "East Asia (excluding China)",
```

```
            "European Union (27)",
```

```
            "Total (excluding China)",
```

```
            "United Kingdom",
```

```
            "United States",
```

```
        ],
```

```
        ["EAsia-China", "EU (27)", "Total-China", "UK", "USA"],
```

```
    )
```



```

data = data.replace(
    [
        "Milk powder, butter, and cheese",
        "Meat and edible offal",
        "Logs, wood, and wood articles",
        "Fish, crustaceans, and molluscs",
        "Non-food manufactured goods",
        "Mechanical machinery and equip",
        "Electrical machinery and equip",
    ],
    [
        "Dairy",
        "Meat",
        "Wood",
        "Fish",
        "Other Goods",
        "Mech Machines",
        "E-Machines",
    ],
)

```

# Grouping the data based on Country and Commodity while calculating the sum of Value and resetting the index

```
data = data.groupby(["Country", "Commodity"])["Value"].sum().reset_index()
```

# Passing the data into the SQL Database

```
data.to_sql("top5_products", con=engine, if_exists="replace", index=False)
```

```

# Group data by country and commodity, and calculate total Value
top5_products = data.groupby(["Country", "Commodity"])["Value"].sum()

# Exporting the data into .csv files
top5_products.to_csv("top_5_products.csv")

# Plot the top 5 commodities for each country
countries = top5_products.index.levels[0]
num_plots = len(countries)
num_rows = 3
num_cols = -(-num_plots // num_rows)
fig, axs = plt.subplots(
    num_rows,
    num_cols,
    figsize=(9.39, 6.48),
    num="5 Most Profitable Products",
)
for i, country in enumerate(countries):
    top_commodities = top5_products.loc[country].nlargest(5)
    num_commodities = min(6, len(top_commodities))
    top_commodities = top_commodities[:num_commodities]
    row = i // num_cols
    col = i % num_cols
    ax = axs[row, col]
    ax.set_ylabel("Value ($)")
    ax.set_title(country)
    ax.bar(top_commodities.index, top_commodities.values)

```

```
ax.tick_params(axis="x", rotation=90)
```

```
fig.suptitle(  
    "Top 5 Most Profitable Products per Country", fontweight="bold", fontsize=16  
)  
mngr = plt.get_current_fig_manager()  
mngr.window.geometry("+661+210")  
plt.tight_layout()  
plt.show()
```

```
def eighth_figure(self):  
    plt.close("all")  
    usecols = [  
        "Direction",  
        "Date",  
        "Commodity",  
        "Measure",  
        "Value",  
        "Weekday",  
    ] # Define the Used Columns for this Graph  
    data = pd.read_csv(  
        "data.csv", usecols=usecols  
    ) # Read the .csv file and save it  
    data = data[(data["Direction"] == "Exports") & (data["Measure"] == "$")]  
  
    # Make the data names Shorter  
    data = data.replace(  

```

```

[
    "Milk powder, butter, and cheese",
    "Meat and edible offal",
    "Logs, wood, and wood articles",
    "Fish, crustaceans, and molluscs",
    "Non-food manufactured goods",
    "Mechanical machinery and equip",
    "Electrical machinery and equip",
],
[
    "Dairy",
    "Meat",
    "Wood",
    "Fish",
    "Other Goods",
    "Mech Machines",
    "E-Machines",
],
)

best_days = data.loc[
    data.groupby("Commodity")["Value"].idxmax()
] # Grouping while Identifying the max

best_days_per_product = best_days.loc[:, ["Commodity", "Date", "Value"]]

# Passing the data into the SQL Database

```

```

best_days_per_product.to_sql(
    "best_days_per_product", con=engine, if_exists="replace", index=False
)

# Exporting the data into .csv files
best_days_per_product.to_csv("best_days_per_product.csv")

num_commodities = len(best_days) # Number of commodities
num_cols = min(num_commodities, 4)
num_rows = (num_commodities + num_cols - 1) // num_cols
fig, axs = plt.subplots(
    num_rows,
    num_cols,
    figsize=(9.39, 6.48),
    num="Most Profitable Day per Product",
)

for i, (commodity, row) in enumerate(best_days.iterrows()):
    ax = axs[i // num_cols, i % num_cols]
    ax.bar(row["Weekday"], row["Value"])
    ax.set_title(row["Commodity"])
    ax.set_ylabel("Value($)")

# Remove any empty plots
for i in range(num_commodities, num_rows * num_cols):
    axs.flat[i].remove()

fig.suptitle("Top Days For Each Commodity", fontweight="bold", fontsize=16)

```

```
mngr = plt.get_current_fig_manager()
mngr.window.geometry("+661+210")
plt.tight_layout(pad=2.0)
plt.show()
```

```
if __name__ == "__main__":
    app = App()
    app.mainloop()
```

# Screenshots Παραδειγμάτων της εφαρμογής και σχήμα Βάσης Δεδομένων

---

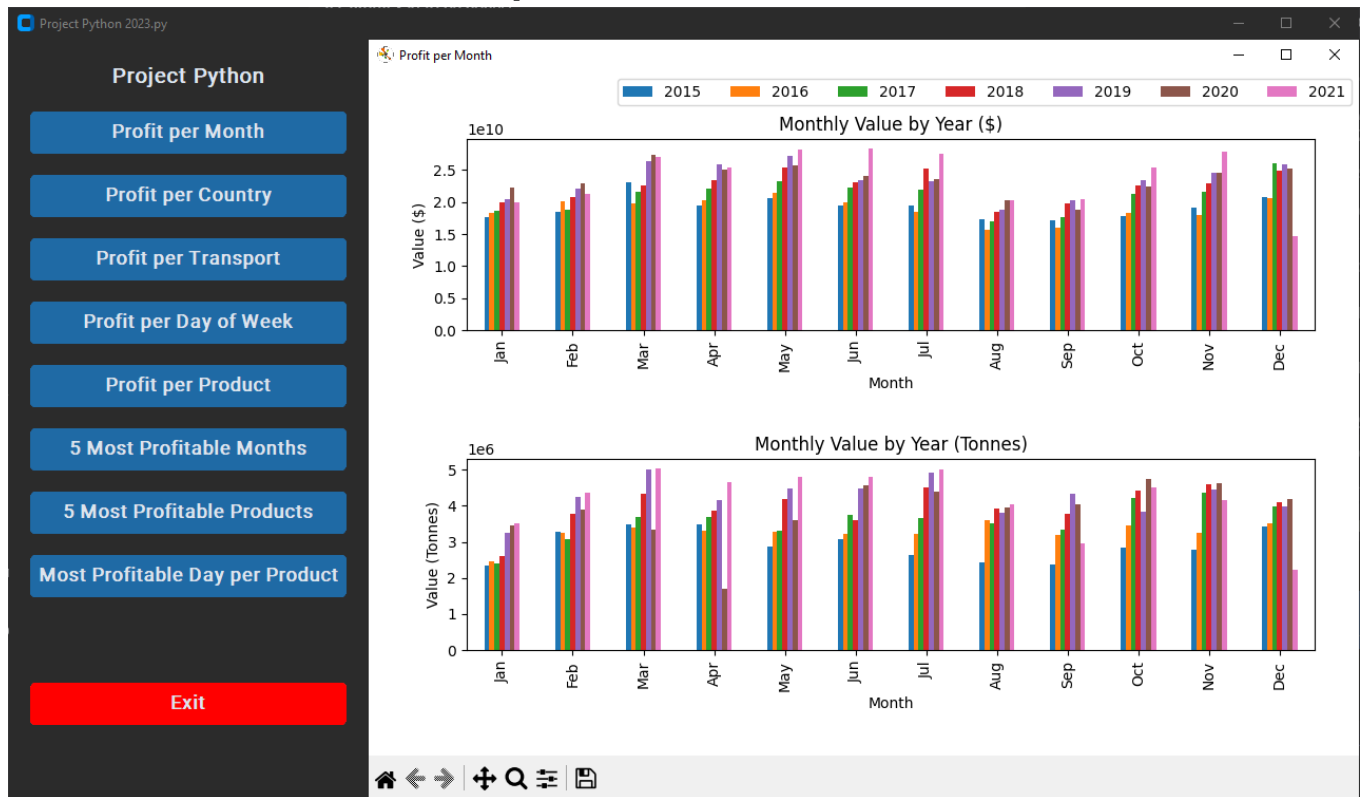
**Το πρόγραμμα ενημερώνει εάν το data.csv υπάρχει στην συσκευή:**

```
[Running] python -u "e:\Jason\Present\Πανεπιστήμιο\Αρχές Γλωσσών Προγραμματισμού και Μεταφραστών\Python-Project-2023\script.py"  
File already exists, skipping download.
```

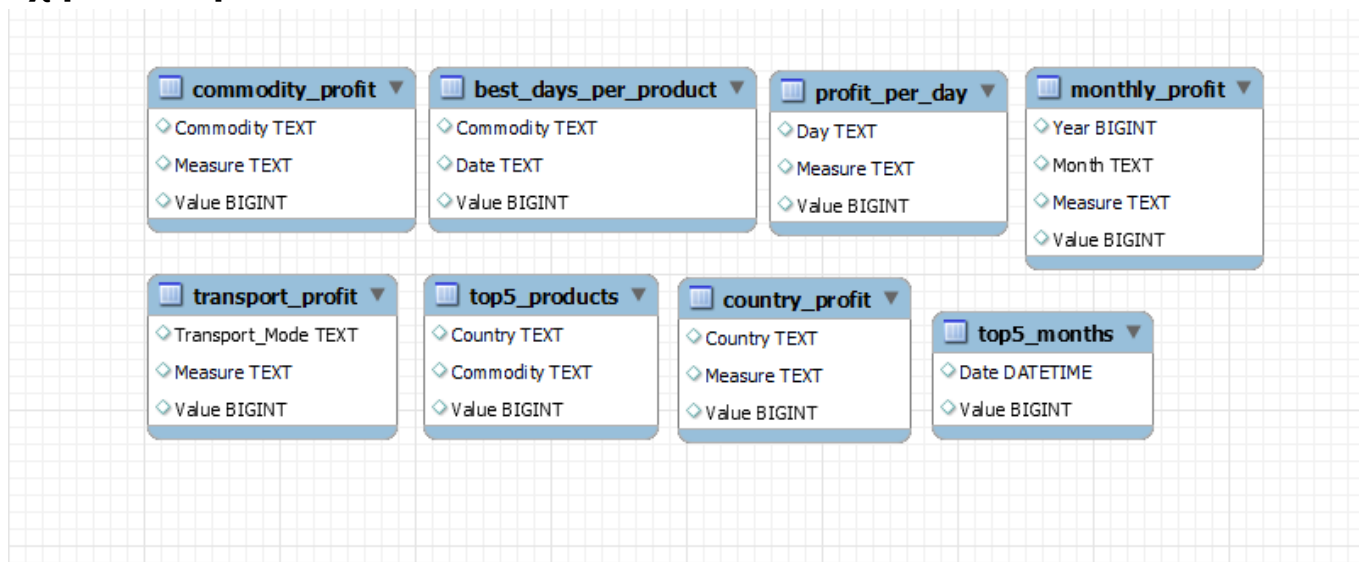
## Αρχικό Παράθυρο GUI



## Πατώντας το Button 'Profit per Month'



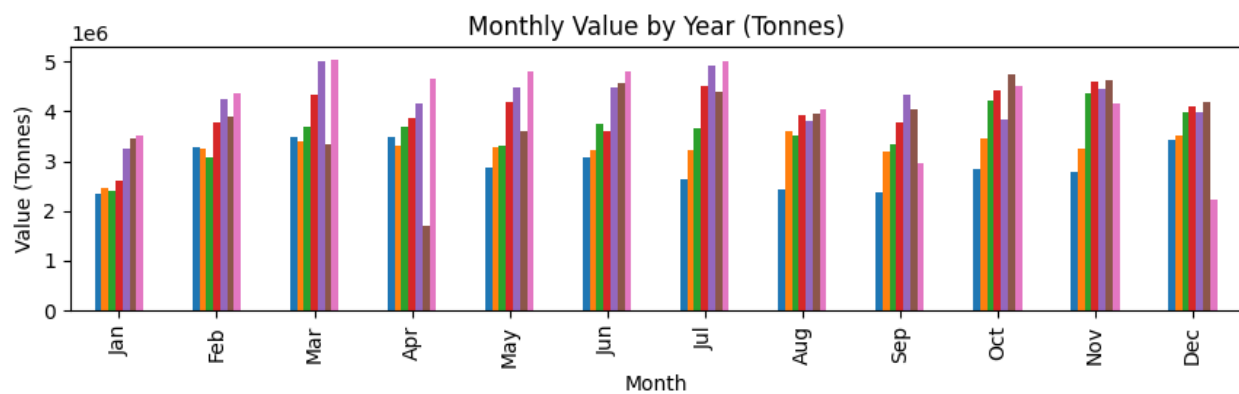
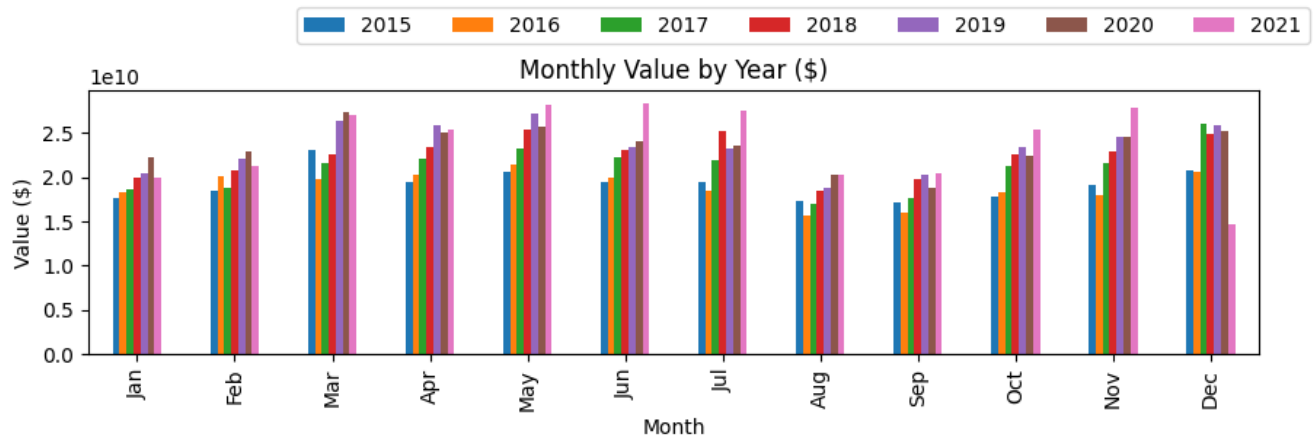
## Σχήμα Βάσης Δεδομένων



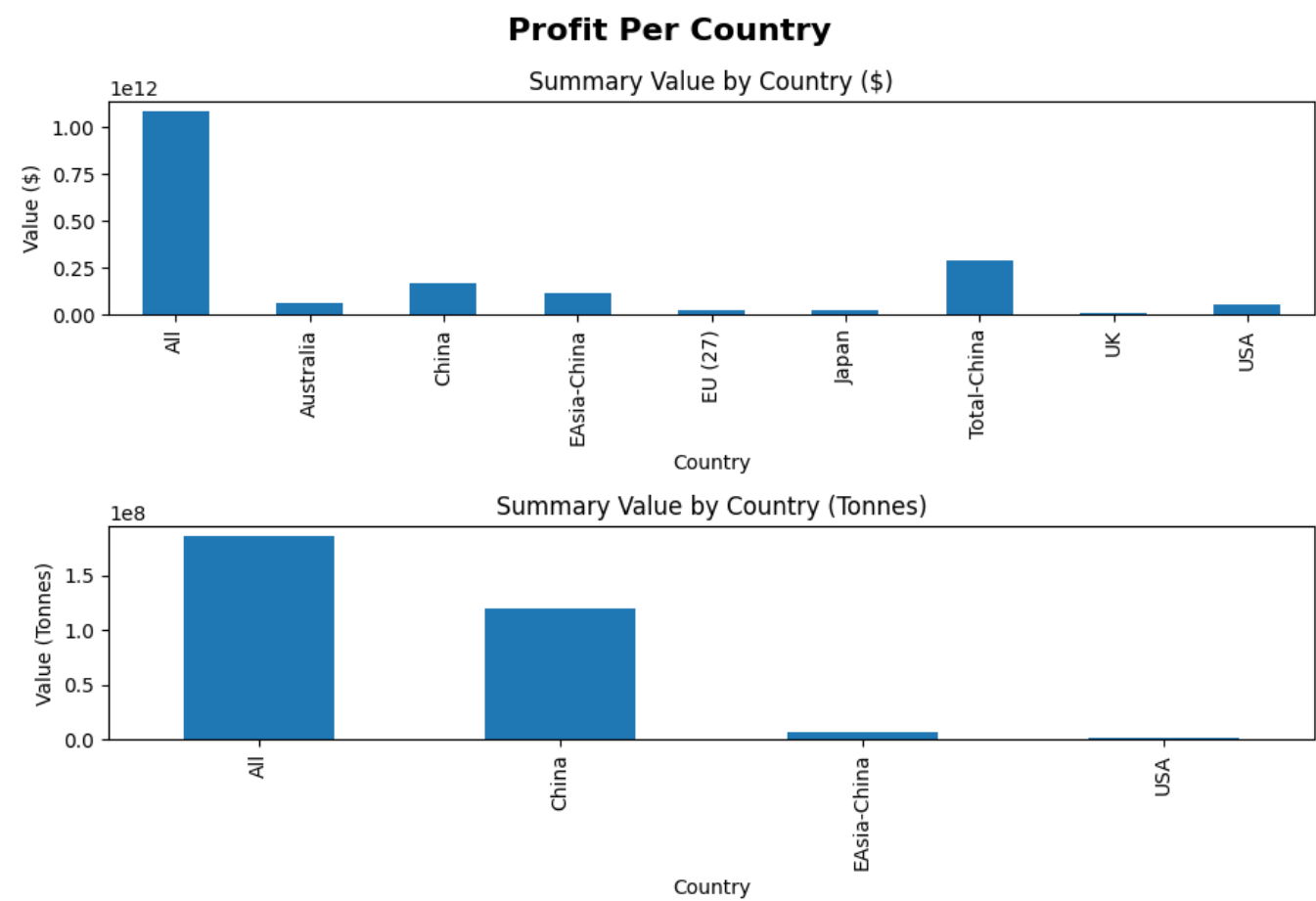


# Ζητούμενα Γραφήματα

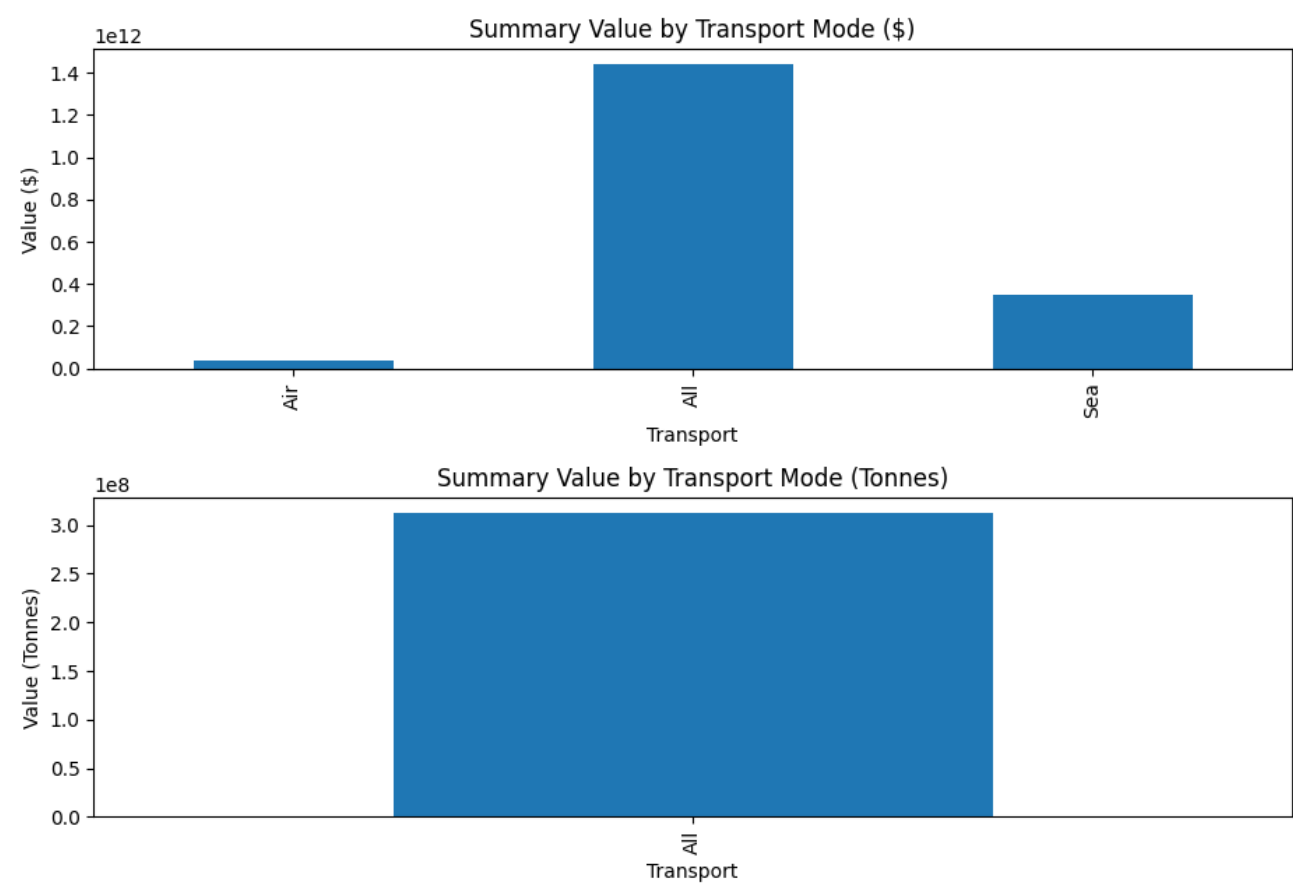
## Συνολική παρουσίαση του τζίρου ανά μήνα



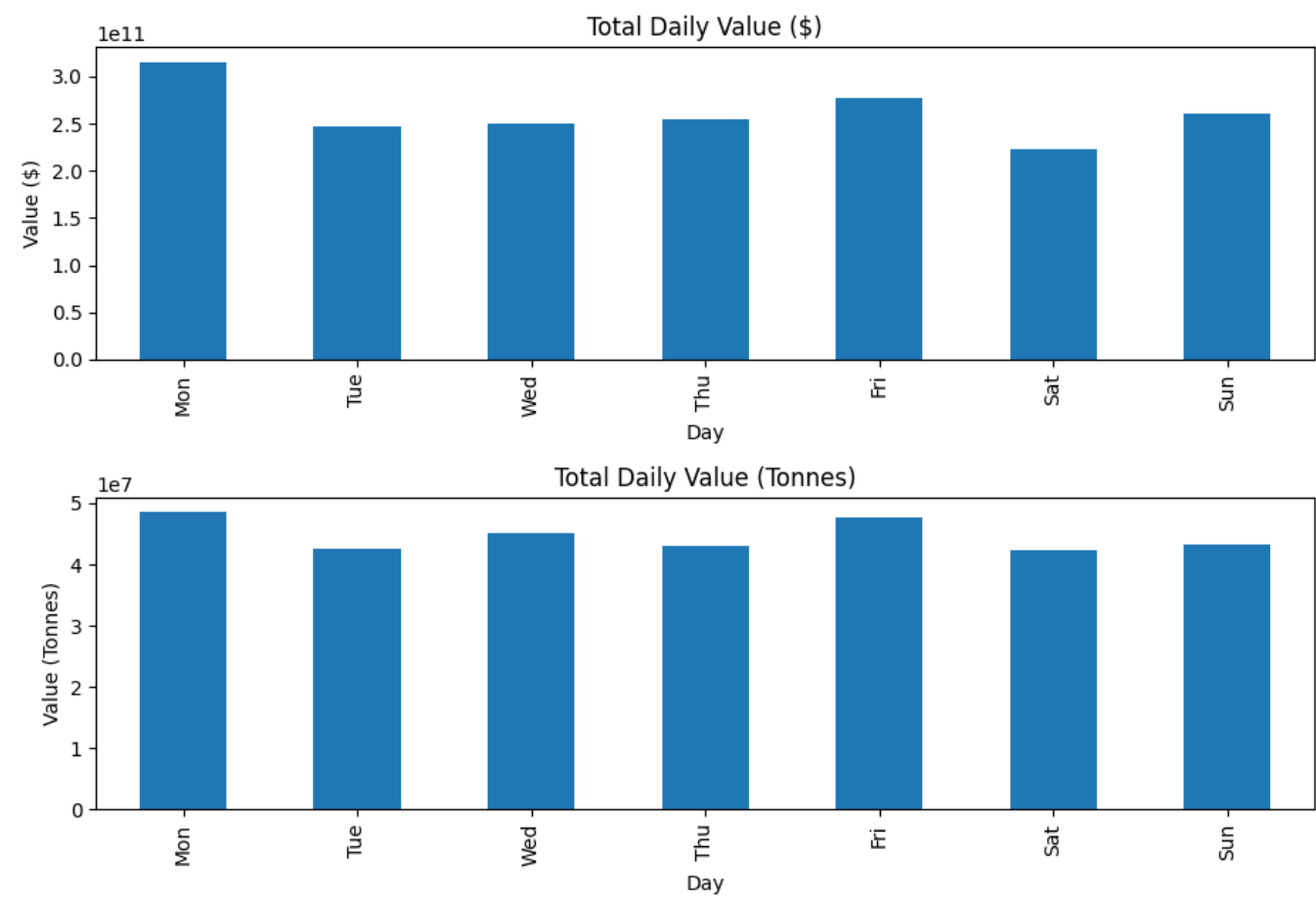
Συνολική παρουσίαση του τζιρου για κάθε χώρα



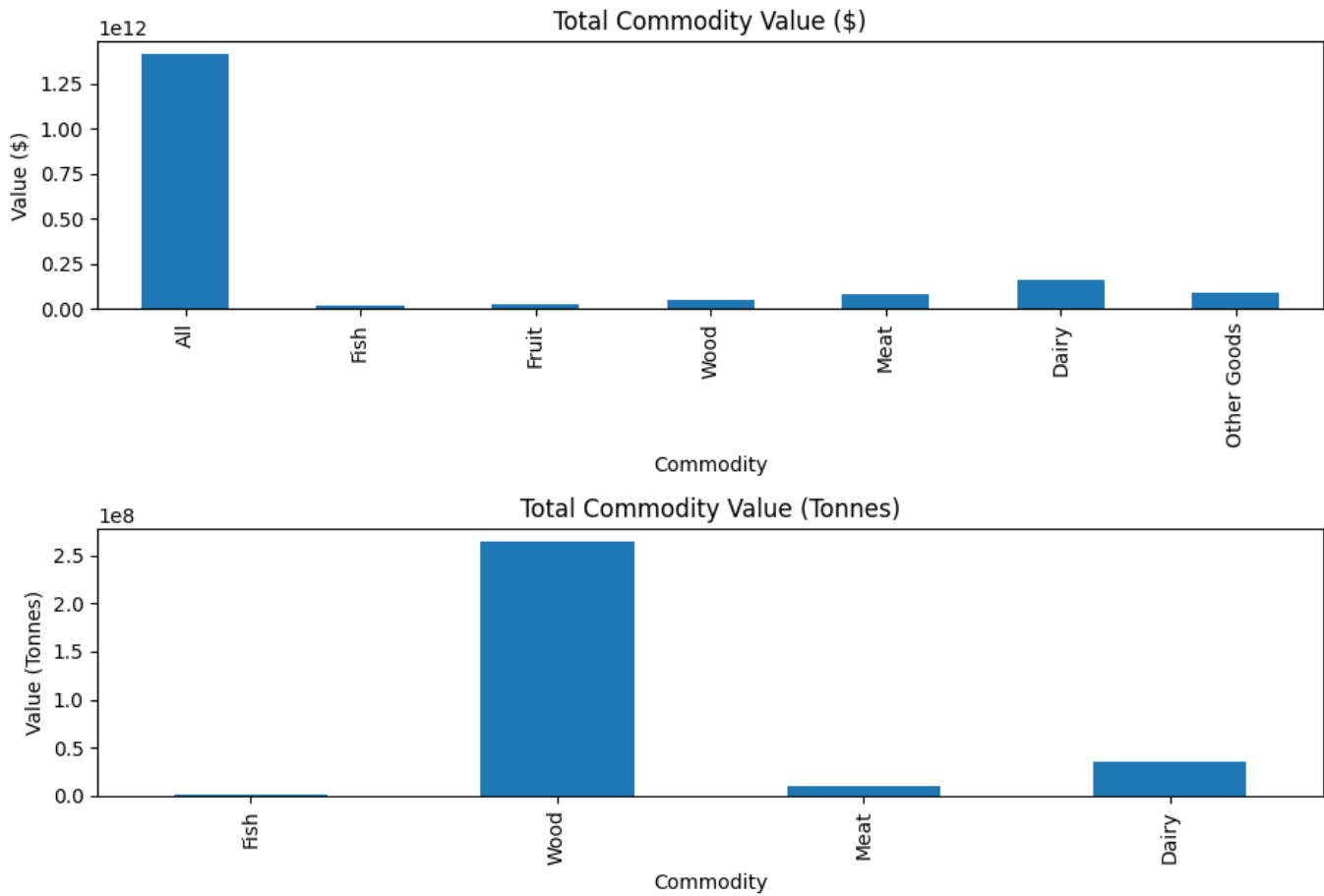
Συνολική παρουσίαση του τζίρου για κάθε μέσο μεταφοράς



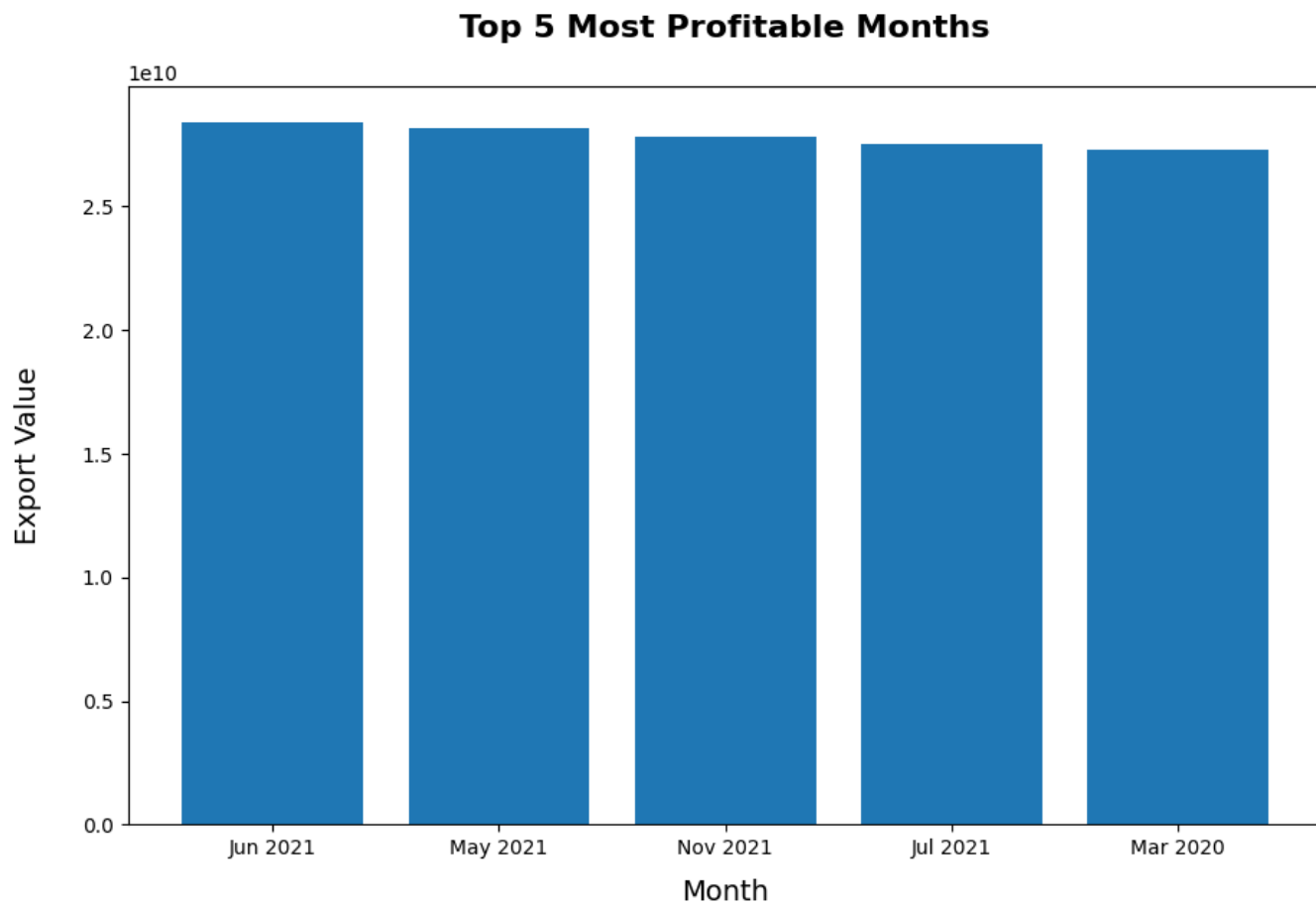
Συνολική παρουσίαση του τζιρού για κάθε μέρα της εβδομάδας



Συνολική παρουσίαση του τζίρου για κάθε κατηγορία εμπορεύματος

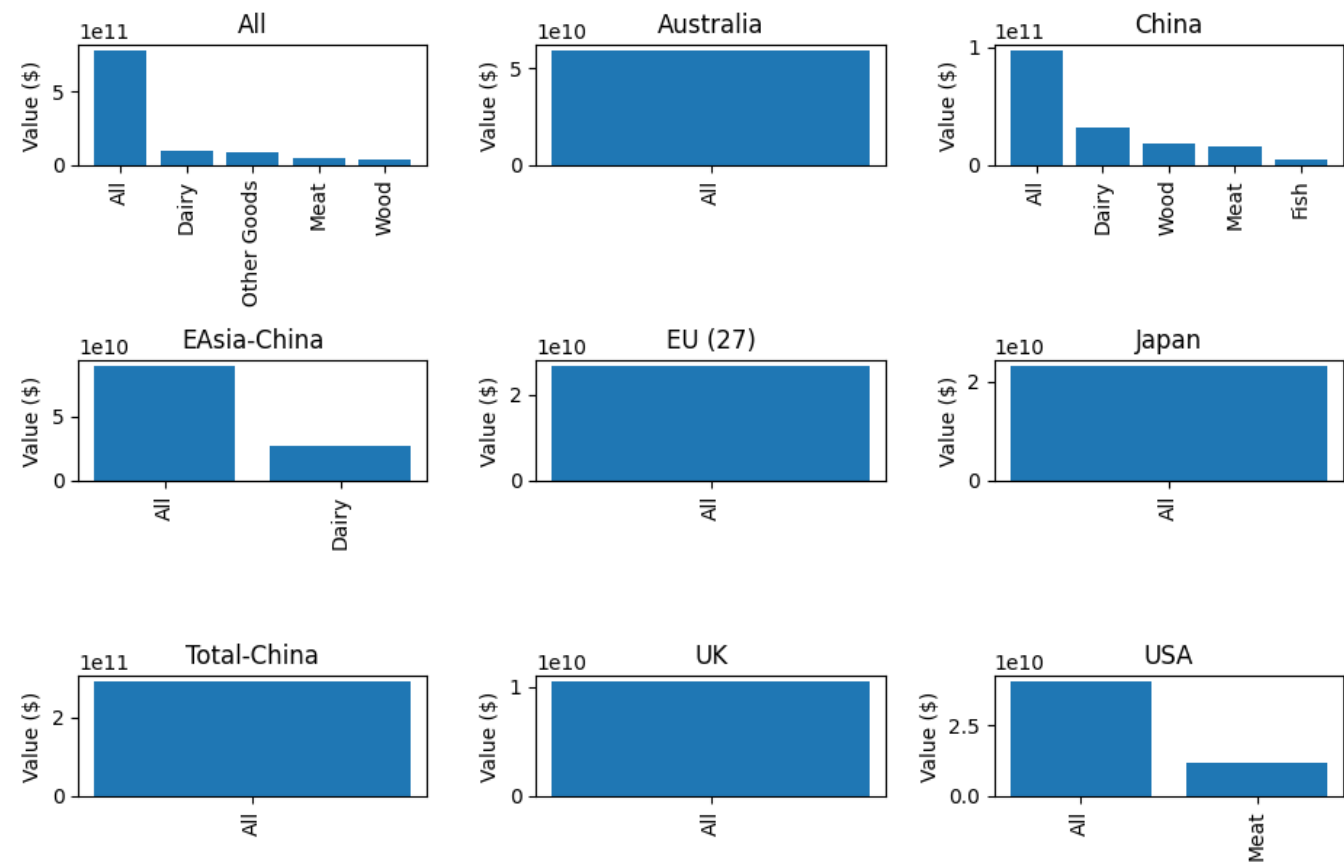


**Παρουσίαση των 5 μηνών με το μεγαλύτερο τζίρο, ανεξαρτήτως μέσου μεταφοράς και είδους ανακυκλώσιμων ειδών**



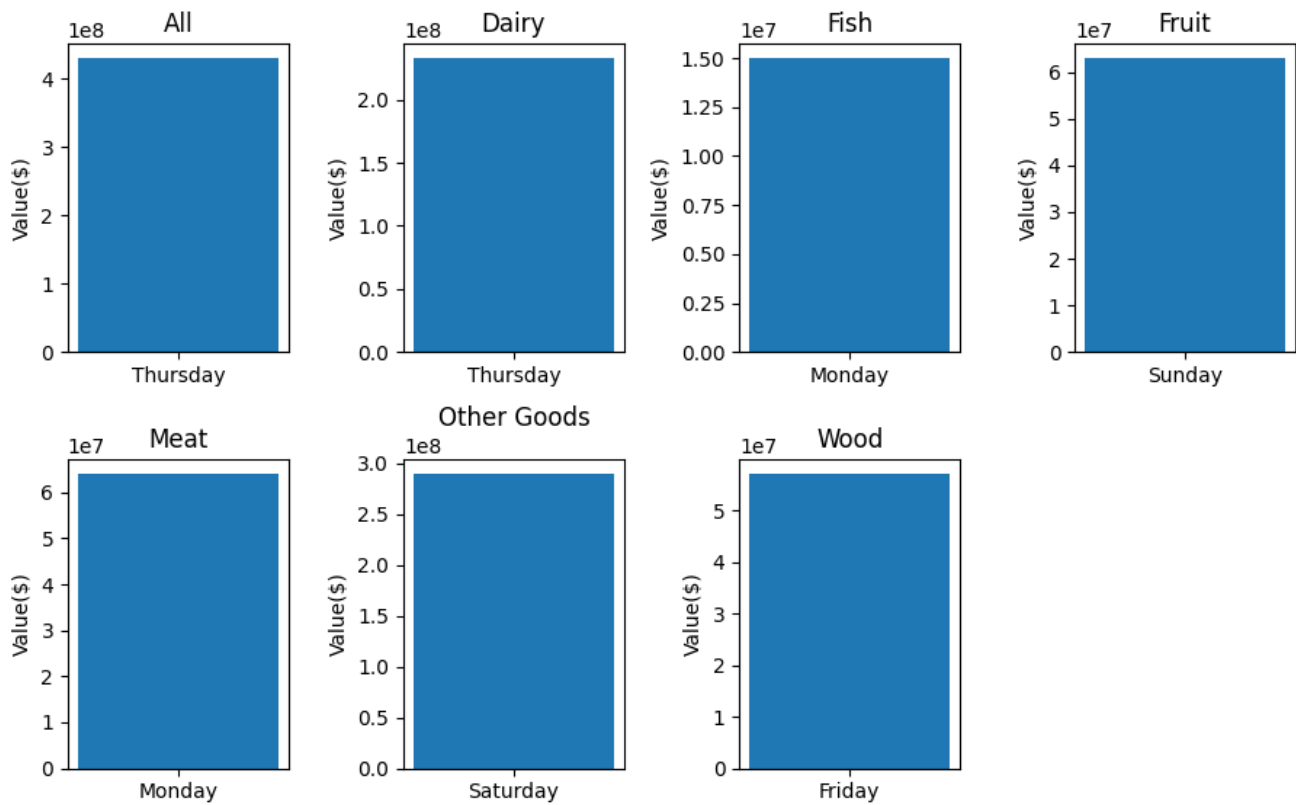
Παρουσίαση των 5 κατηγοριών εμπορευμάτων με το μεγαλύτερο τζίρο, για κάθε χώρα

Top 5 Most Profitable Products per Country



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

Top Days For Each Commodity





# Σχόλια-Παραδοχές Υλοποίησης

---

Για την υλοποίηση του Project και των ζητούμενων, θεώρησα τα εξής:

- Για να μπορέσετε να τρέξετε το Script της Python θα χρειαστεί να τρέξετε τα εξής commands στο command line:
  1. `pip install customtkinter` (Για την βιβλιοθήκη customtkinter)
  2. `pip install pandas` (Για την βιβλιοθήκη pandas)
  3. `pip install matplotlib` (Για την βιβλιοθήκη matplotlib)
  4. `pip install sqlalchemy` (Για την βιβλιοθήκη sqlalchemy)
- Για την βελτίωση της ταχύτητας φόρτωσης των δεδομένων κατά την δημιουργία των διαγραμμάτων, χρησιμοποίηώ κάθε φορά μόνο τις στήλες με τα δεδομένα που μου χρειάζονται.
- Τα κέρδη υπολογίζονται μόνο από τα δεδομένα των γραμμών των οποίων η στήλη "Direction" έχει την τιμή "Export", με την λογική ότι τα Έσοδα υπολογίζονται από την Εξαγωγή Προϊόντων.
- Τα ονόματα των ημερών, μηνών, χωρών και των τροφίμων, έχουν αλλάξει με της ίδιας σημασίας ονομασίες, αλλά πιο μικρές, ώστε τα διαγράμματα να είναι πιο ευανάγνωστα.
- Η μέγιστη βαρύτητα δόθηκε στην λειτουργικότητα του κώδικα, στην ακρίβεια των δεδομένων και στην εμφάνιση του προγράμματος για να είναι φιλικό προς τον χρήστη.
- Τέλος, για την βάση δεδομένων, δημιούργησα 8 πίνακες (όσοι και τα ζητούμενα), οποίοι δεν έχουν καμία σύνδεση μεταξύ τους και είναι απλά για την αποθήκευση των επεξεργασμένων δεδομένων.

# Βιβλιογραφία Παραπομπές

---

- GitHub Link for customtkinter: <https://github.com/TomSchimansky/CustomTkinter>