|  |  |
| --- | --- |
| **Python Project 202****3** |  |

# Στοιχεία Φοιτητή

Όνομα: Ιάσονας

Επώνυμο: Παυλόπουλος

ΑΜ: 1084565

Έτος: 3ο

Email: up1084565@upnet.grΕικόνα που περιέχει κείμενο

Περιγραφή που δημιουργήθηκε αυτόματα

# Περιεχόμενα

1. [**Εισαγωγή**](#_Εισαγωγή)
2. **[Κώδικας Python](#_Κώδικας_Python_1)**
3. **[Screenshots Παραδειγμάτων της εφαρμογής και σχήμα Βάσης Δεδομένων](#_Screenshots_Παραδειγμάτων_της)**
4. **[Ζητούμενα Γραφήματα](#_Ζητούμενα_Γραφήματα)**
5. **[Σχόλια-Παραδοχές Υλοποίησης](#_Σχόλια-Παραδοχές_Υλοποίησης)**
6. **[Βιβλιογραφία-Παραπομπές](#_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(["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", values="Value", aggfunc="sum"

)

monthly\_tonnes = monthly\_data[monthly\_data["Measure"] == "Tonnes"].pivot\_table(

index="Month", 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", nrows=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", nrows=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=axs[0], legend=False

)

axs[0].set\_title("Summary Value by Country ($)")

axs[0].set\_xlabel("Country")

axs[0].set\_ylabel("Value ($)")

tonnes\_country.plot(x="Country", y="Value", kind="bar", ax=axs[1], legend=False)

axs[1].set\_title("Summary Value by Country (Tonnes)")

axs[1].set\_xlabel("Country")

axs[1].set\_ylabel("Value (Tonnes)")

fig.suptitle("Profit Per Country", fontweight="bold", fontsize=16)

mngr = plt.get\_current\_fig\_manager()

mngr.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 seperate 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,

)

mngr = plt.get\_current\_fig\_manager()

mngr.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()

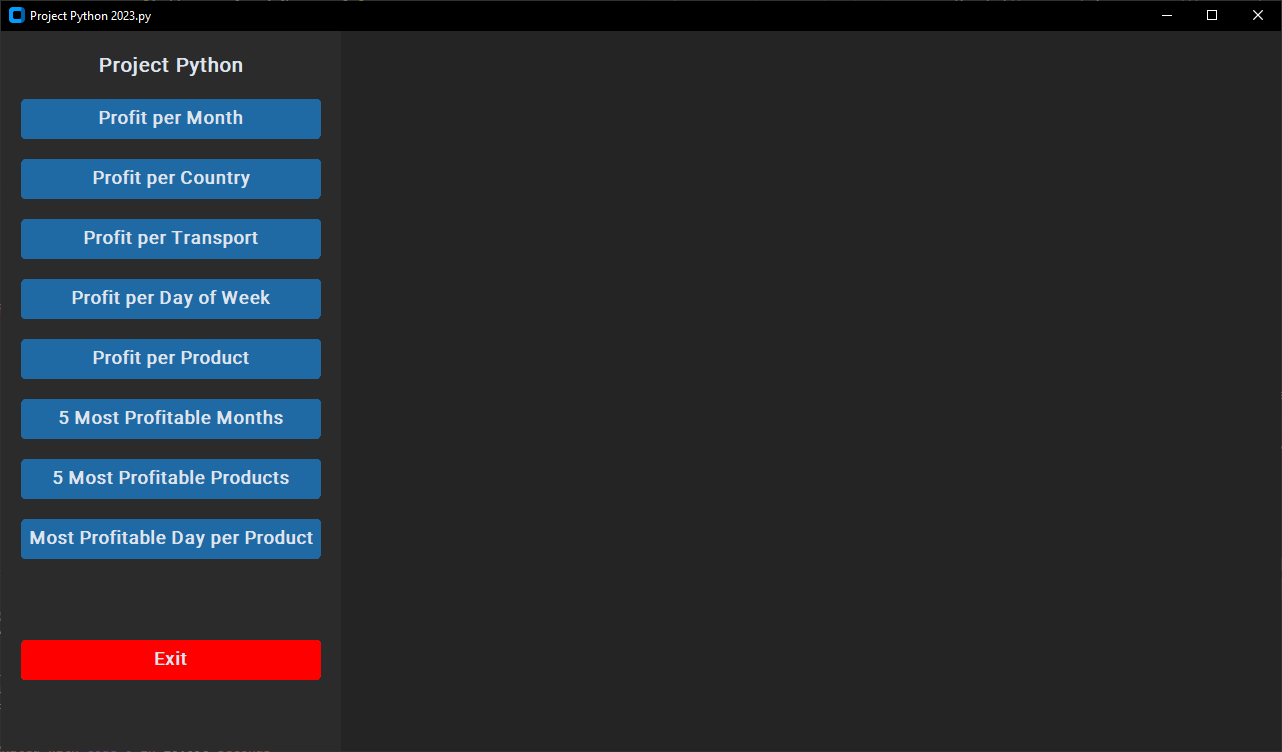
if \_\_name\_\_ == "\_\_main\_\_":

app = App()

app.mainloop()

# Screenshots Παραδειγμάτων της εφαρμογής και σχήμα Βάσης Δεδομένων

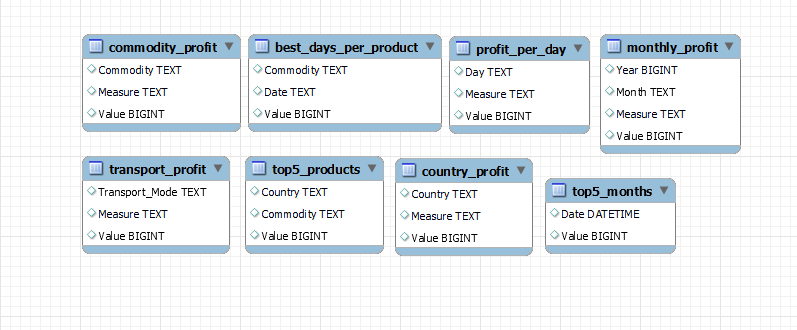
**Το πρόγραμμα ενημερώνει εάν το data.csv υπάρχει στην συσκευή:**

**Αρχικό Παράθυρο GUI**

**Εικόνα που περιέχει διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματαΠατώντας το Button ‘Profit per Month’**

**Σχήμα Βάσης Δεδομένων**



# Ζητούμενα Γραφήματα

**Συνολική παρουσίαση του τζίρου ανά μήνα**

**Εικόνα που περιέχει διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα**

**Εικόνα που περιέχει διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματαΣυνολική παρουσίαση του τζίρου για κάθε χώρα**

**Συνολική παρουσίαση του τζίρου για κάθε μέσο μεταφοράς**

Εικόνα που περιέχει διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα

Εικόνα που περιέχει διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματα**Συνολική παρουσίαση του τζίρου για κάθε μέρα της εβδομάδας**

**Εικόνα που περιέχει διάγραμμα

Περιγραφή που δημιουργήθηκε αυτόματαΣυνολική παρουσίαση του τζίρου για κάθε κατηγορία εμπορεύματοςΠαρουσίαση των 5 μηνών με το μεγαλύτερο τζίρο, ανεξαρτήτως μέσου μεταφοράς και είδους ανακυκλώσιμων ειδών**

Εικόνα που περιέχει διάγραμμα

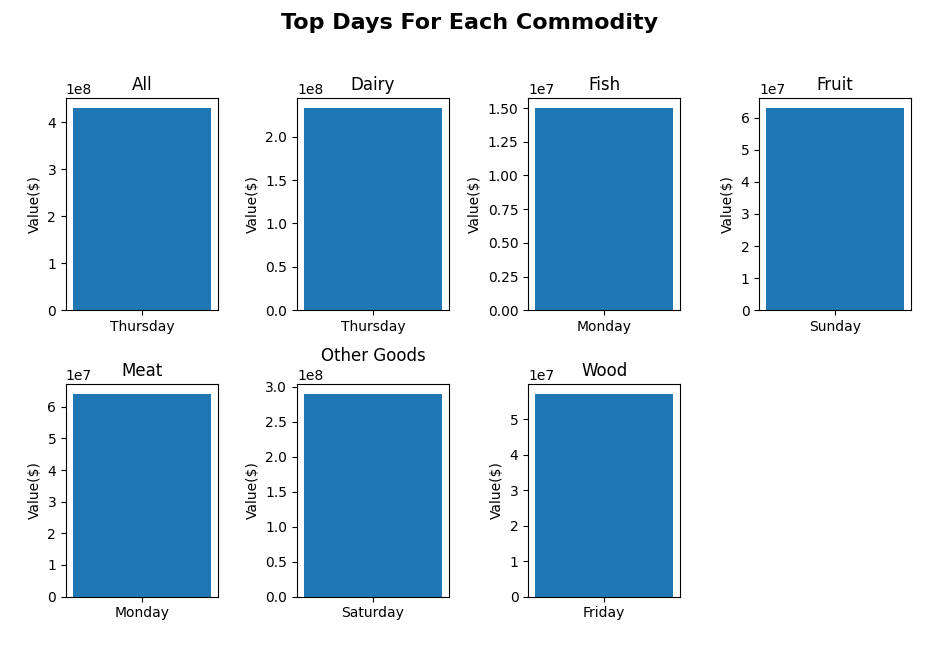
Περιγραφή που δημιουργήθηκε αυτόματα

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

Εικόνα που περιέχει διάγραμμα, σχηματικό

Περιγραφή που δημιουργήθηκε αυτόματα

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



# Σχόλια-Παραδοχές Υλοποίησης

Για την υλοποίηση του Project και των ζητουμένων, 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)
5. pip install mysqlclient (Για τα dependencies της sqlalchemy)

* Για την βελτίωση της ταχύτητας φόρτωσης των δεδομένων κατά την δημιουργία των διαγραμμάτων, χρησιμοποιώ 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), οι 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