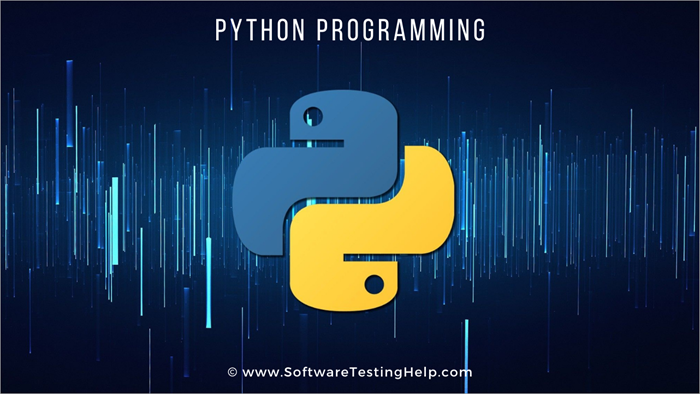


***Inove Coding School***

Proyecto Final

*“Python Analytics”*

***Alumno:***

**Valentín Marco Imperio**

***Profesores:***

**Johana Rangel**

**Julián Salinas**

***Fecha de Entrega:***

**28/01/2023**

***Introducción***

El objetivo del presente proyecto final es aplicar los conocimientos aprendidos a lo largo del curso *Programador Python*. Particularmente se trabajará con los siguientes conceptos:

* Funciones y variables.
* Ciclos y bucles *for*.
* Listas, Tuplas y Diccionarios.
* Comprensión de Listas.
* Arrays y la librería Numpy.
* Bases de datos.
* SQL ORM con SQL Alchemy.
* Modularización.
* Gráficos con Matplotlib.
* Utilización de las Buenas Prácticas de la Programación.

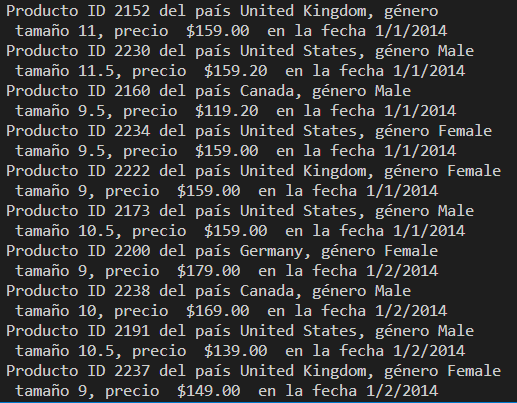
Dado que el tema elegido es el análisis de un dataset de ventas de calzado internacionales, se explicará el proyecto y su aplicación en el próximo apartado.

***Desarrollo***

Se explica a continuación en orden cronológico el objetivo de cada función utilizada.

***Función read\_db()***

La misma se utiliza para la limpieza, ordenamiento y lectura de los datos.

  
**Figura 1** *– Primeros 10 datos de la tabla Original – Screen Python*

*Texto

Descripción generada automáticamente***Figura 2** *– Primeros 10 datos de la tabla Filtrada – Screen Python*

Por ejemplo, al comparar la Figura 1 con la Figura 2, se observa que el producto 2152 fue removido por tener un dato faltante en “gender”.

***Función paises\_unicos()***

Esta función sirve para obtener una lista de todos los países en los que se efectuó al menos una venta.

 **Figura 3** – *Lista de países*

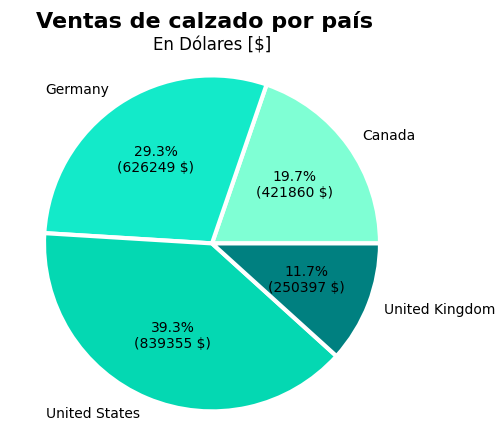
***Función ventas\_pais()***

La misma sirve para obtener un diccionario de todos los países con la cantidad de ventas en dólares.



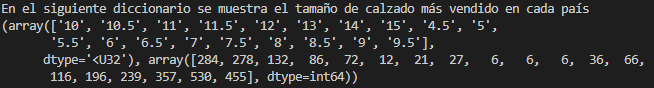
**Figura 4** – Diccionario con v*entas en dólares por país*

Se muestra el resultado obtenido en el siguiente gráfico de tortas.



***Función calzado\_pais()***

Esta función sirve para obtener un diccionario del tamaño de calzado más vendido en cada país.



**Figura 4** – *Lista de arrays con el N° de ventas por tamaño de calzado en Canadá*

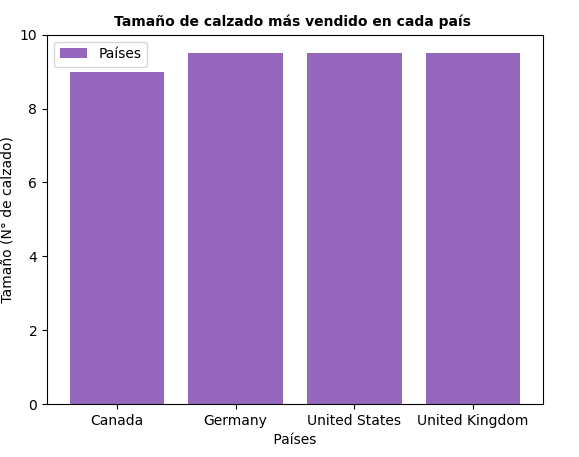
Se obtienen la cantidad de ventas según el tamaño de calzado para cada país, en este ejemplo se utilizó Canadá. Efectivamente se obtiene que tamaño más comercializado es el calzado N°9, con 530 ventas realizadas.

Luego en la figura 5 se muestran el resultado para cada país:



**Figura 5** – *Diccionario con el N° de calzado más vendido por país*

Se muestra el resultado obtenido en el siguiente gráfico de barras.



***Función ventas\_genero\_pais()***

En esta función el usuario puede elegir un género target y de acuerdo al mismo se muestran las ventas de calzado por país. A continuación, se pueden ver dos ejemplos, para *Female* y para *Male*.

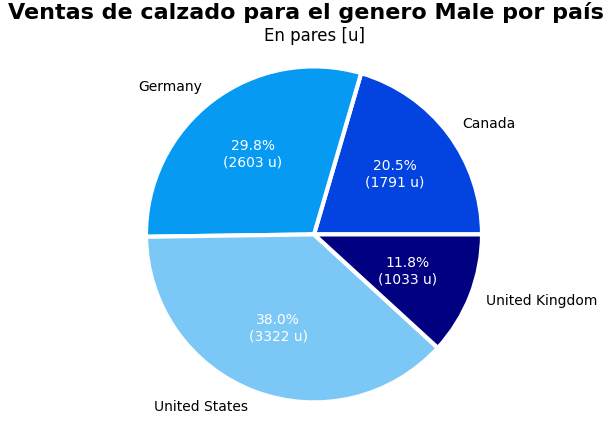


**Figura 6** – *Diccionario con las ventas de calzado para mujeres en cada país*



**Figura 7** – *Diccionario con las ventas de calzado para hombres en cada país*

Se muestra el resultado obtenido en el siguiente gráfico de tortas.



***Bonus Track: Gráficos***

Se elaboraron distintos gráficos para cada función utilizando la librería *Matplotlib*. Los mismos se encuentran con su función correspondiente dentro de cada apartado.

***Anexo: Código***

Se adjunta el código utilizado para el llevado a cabo del proyecto correspondiente al archivo main.py y graficos.py

#######################################################################

# Proyecto Final - Inove Escuela de Código

# Tema Elegido: Python Analytics

# Alumno: Valentín Imperio.

# Versión: 1.0

# Profesores: Johana Rangel y Julián Salinas.

# Curso: Programador Python.

# Fecha de Entrega: 28/01/2023

# ARCHIVO PRINCIPAL: MAIN.PY

#######################################################################

# DESARROLLO DEL PROYECTO #

#### Librerías utilizadas ###

import numpy as np

import re # Para convertir a float y remover $

import sqlalchemy

from sqlalchemy import Column, Integer, String # Herramientas para crear la tabla.

from sqlalchemy.ext.declarative import declarative\_base # Molde para tablas de bases de datos.

from sqlalchemy.orm import sessionmaker # Sirve para poder iniciar la sesión

### Creación del motor (engine) de la base de datos

engine = sqlalchemy.create\_engine("sqlite:///ventas\_calzados.db")

base = declarative\_base()

###### Creación de la Clase Ventas ######

# Para armar la estructura de la tabla dada (visualizador de bases de datos en

# https://extendsclass.com/sqlite-browser.html) con el ORM de SqlAlchemy se

# genera la siguiente clase.

class Ventas(base):

    \_\_tablename\_\_ = "venta"

    id = Column(Integer , primary\_key=True)

    date = Column(String,nullable=True)

    product\_id = Column(Integer,nullable=True)

    country = Column(String,nullable=True)

    gender = Column(String,nullable=True)

    size = Column(String,nullable=True)

    price = Column(String,nullable=True)

    def \_\_repr\_\_(self): # Método informativo

        return f"Producto ID {self.product\_id} del país {self.country}, género {self.gender} \

        \n tamaño {self.size}, precio {self.price} en la fecha {self.date}"

####### Funciones #######

# Función READ\_DB: Para leer la base de datos ventas\_calzado.db y organizar los datos.

def read\_db(limit=10):

    # Crear la sesión

    Sesion = sessionmaker(bind=engine)

    sesion = Sesion()

    # Buscar todas las ventas

    query = sesion.query(Ventas)

    # Aplica el limite si está definido

    if limit > 0:

        query1 = query.limit(limit)

    # Leer una venta a la vez e imprime en pantalla

    print("Estos son los primeros 10 datos de la tabla ORIGINAL")

    for venta in query1:

        print(venta)

    nro\_filas=query.count()

    print("La tabla ORIGINAL tiene",nro\_filas,"filas.") #14967

    # Se eliminan las filas que tengan un valor nulo

    print("\n")

    print("Removiendo filas con valores vacíos...")

    print("\n")

    # Estas son las filas con valores nulos.

    query2 = sesion.query(Ventas).filter(

        (Ventas.date == "") |

        (Ventas.product\_id == "") |

        (Ventas.country == "") |

        (Ventas.gender == "") |

        (Ventas.size == "") |

        (Ventas.price == "")

        )

    nro\_filas2 = query2.count()

    print("Se eliminaran",nro\_filas2,"filas de la tabla por datos faltantes") #120

    print("\n")

    # Se procede a borrar las filas mencionadas.

    sesion.query(Ventas).filter(

        (Ventas.date == "") |

        (Ventas.product\_id == "") |

        (Ventas.country == "") |

        (Ventas.gender == "") |

        (Ventas.size == "") |

        (Ventas.price == "")

        ).delete()

    sesion.commit() # Para guardar los cambios.

    # Se vuelve a hacer la consulta para ver la tabla filtrada

    query = sesion.query(Ventas)

    # Aplica el limite si está definido

    if limit > 0:

        query1 = query.limit(limit)

    print("Estos son los primeros 10 datos de la tabla FILTRADA")

    for venta in query1:

        print(venta)

    nro\_filas3 = query.count()

    print("Quedan",nro\_filas3,"filas en la tabla.") #14847

    query3 = sesion.query(Ventas)

    country = np.array([])

    gender = np.array([])

    size = np.array([])

    price = np.array([])

    for i in query3:

        country = np.append(country, i.country)

        gender = np.append(gender,i.gender)

        size = np.append(size,i.size)

        price\_str = " ".join(i.price)

        price\_i = float(re.sub(r'[^\d\-.]', '', price\_str)) # Sacar $ y pasar a float

        price = np.append(price,price\_i)

    return country, gender, size, price

# Función PAÍSES ÚNICOS: Para ver en qué países hubo ventas:

def paises\_unicos(country):

    paises\_unicos = np.unique(country)

    print(paises\_unicos)

def ventas\_pais(countries, country, price):

    # Crear unas máscaras para obtener los índices del array

    # que corresponden a los paises

    mask\_0 = country == countries[0] #"Canada"

    mask\_1 = country == countries[1] #"Germany"

    mask\_2 = country == countries[2] #"United States"

    mask\_3 = country == countries[3] #"United Kingdom"

    # Utilizo las máscaras para acceder a los indices del array price

    # que corresponden a cada pais

    ventas\_canada = price[mask\_0]

    ventas\_germany = price[mask\_1]

    ventas\_united\_states = price[mask\_2]

    ventas\_united\_kingdom = price[mask\_3]

    suma\_canada = np.sum(ventas\_canada)

    suma\_germany = np.sum(ventas\_germany)

    suma\_united\_states = np.sum(ventas\_united\_states)

    suma\_united\_kingdom = np.sum(ventas\_united\_kingdom)

    Countries\_dict = {}

    Countries\_dict.update({ countries[0]: suma\_canada, countries[1]: suma\_germany, countries[2]: suma\_united\_states, countries[3]: suma\_united\_kingdom})

    return(Countries\_dict)

# Función CALZADO PAÍS: Para ver qué tamaño fue el más vendido en cada país.

def calzado\_pais(countries, country, size):

    # Crear unas máscaras para obtener los índices del array

    # que corresponden a los paises

    mask\_0 = country == countries[0] #"Canada"

    mask\_1 = country == countries[1] #"Germany"

    mask\_2 = country == countries[2] #"United States"

    mask\_3 = country == countries[3] #"United Kingdom"

    # Utilizo las máscaras para acceder a los indices del array size

    # que corresponden a cada pais

    tamaño\_canada = size[mask\_0]

    tamaño\_germany = size[mask\_1]

    tamaño\_united\_states = size[mask\_2]

    tamaño\_united\_kingdom = size[mask\_3]

    sizes\_canada = np.unique(tamaño\_canada,return\_counts=True)

    sizes\_germany = np.unique(tamaño\_germany,return\_counts=True)

    sizes\_united\_states = np.unique(tamaño\_united\_states,return\_counts=True)

    sizes\_united\_kingdom = np.unique(tamaño\_united\_kingdom,return\_counts=True)

    mask\_mas\_vendido\_C = sizes\_canada[1] == max(sizes\_canada[1])

    tamaño\_mas\_vendido\_C = sizes\_canada[0][mask\_mas\_vendido\_C]

    tamaño\_mas\_vendido\_C = " ".join(tamaño\_mas\_vendido\_C .tolist()) # Con tolist() se remuev el dtype y con el join se pasa de lista a str

    mask\_mas\_vendido\_G = sizes\_germany[1] == max(sizes\_germany[1])

    tamaño\_mas\_vendido\_G = sizes\_germany[0][mask\_mas\_vendido\_G]

    tamaño\_mas\_vendido\_G = " ".join(tamaño\_mas\_vendido\_G .tolist())

    mask\_mas\_vendido\_USA = sizes\_united\_states[1] == max(sizes\_united\_states[1])

    tamaño\_mas\_vendido\_USA = sizes\_united\_states[0][mask\_mas\_vendido\_USA]

    tamaño\_mas\_vendido\_USA = " ".join(tamaño\_mas\_vendido\_USA .tolist())

    mask\_mas\_vendido\_UK = sizes\_united\_kingdom[1] == max(sizes\_united\_kingdom[1])

    tamaño\_mas\_vendido\_UK = sizes\_united\_kingdom[0][mask\_mas\_vendido\_UK]

    tamaño\_mas\_vendido\_UK = " ".join(tamaño\_mas\_vendido\_UK .tolist())

    Sizes\_dict = {}

    Sizes\_dict.update({ countries[0]: tamaño\_mas\_vendido\_C, countries[1]: tamaño\_mas\_vendido\_G, countries[2]: tamaño\_mas\_vendido\_USA, countries[3]: tamaño\_mas\_vendido\_UK})

    return(Sizes\_dict)

# Función VENTAS GENERO PAÍS: Para ver qué genero compro la mayor cantidad de calzado en cada país.

def ventas\_genero\_pais(countries, gender\_target, country, gender):

    generos\_unicos = np.unique(gender) # "Female", "Male" y "Unix"

    # Crear unas máscaras para obtener los índices del array

    # que corresponden a los paises

    mask\_0 = country == countries[0] #"Canada"

    mask\_1 = country == countries[1] #"Germany"

    mask\_2 = country == countries[2] #"United States"

    mask\_3 = country == countries[3] #"United Kingdom"

    # Utilizo las máscaras para acceder a los indices del array size

    # que corresponden a cada pais

    genero\_canada = gender[mask\_0]

    genero\_germany = gender[mask\_1]

    genero\_united\_states = gender[mask\_2]

    genero\_united\_kingdom = gender[mask\_3]

    gender\_canada = np.unique(genero\_canada,return\_counts=True)

    gender\_germany = np.unique(genero\_germany,return\_counts=True)

    gender\_united\_states = np.unique(genero\_united\_states,return\_counts=True)

    gender\_united\_kingdom = np.unique(genero\_united\_kingdom,return\_counts=True)

    mask\_target\_C = gender\_canada[0] == gender\_target

    genero\_target\_C = gender\_canada[1][mask\_target\_C]

    genero\_target\_C = genero\_target\_C .tolist()[0] # Con tolist() se remuev el dtype y con el [0] se obtiene el unico elemento de la lista.

    mask\_target\_G = gender\_germany[0] == gender\_target

    genero\_target\_G = gender\_germany[1][mask\_target\_G]

    genero\_target\_G = genero\_target\_G .tolist()[0]

    mask\_target\_USA = gender\_united\_states[0] == gender\_target

    genero\_target\_USA = gender\_united\_states[1][mask\_target\_USA]

    genero\_target\_USA = genero\_target\_USA .tolist()[0]

    mask\_target\_UK = gender\_united\_kingdom[0] == gender\_target

    genero\_target\_UK = gender\_united\_kingdom[1][mask\_target\_UK]

    genero\_target\_UK = genero\_target\_UK .tolist()[0]

    Target\_Genre\_dict = {}

    Target\_Genre\_dict.update({ countries[0]: genero\_target\_C, countries[1]: genero\_target\_G, countries[2]: genero\_target\_USA, countries[3]: genero\_target\_UK})

    return(Target\_Genre\_dict)

####### Bloque principal del programa #######

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

    data = read\_db()

    country = data[0]

    gender = data[1]

    size = data[2]

    price = data[3]

    countries = ["Canada","Germany","United States", "United Kingdom"]

    gender\_target = "Male" # Las opciones a analizar son: "Female", "Male" y "Unix"

    print("En el siguiente array se muestran los países en los que hubo ventas:")

    paises\_unicos(country)

    print("En el siguiente diccionario se muestra el total de ventas por país:")

    print(ventas\_pais(countries, country, price))

    print("En el siguiente diccionario se muestra el tamaño de calzado más vendido en cada país:")

    print(calzado\_pais(countries, country, size))

    print("En el siguiente diccionario se muestan las ventas para el género "+ gender\_target+ " en cada país:")

    print(ventas\_genero\_pais(countries, gender\_target, country, gender))

    print("Termine! Que buen programa soy ^^")

#######################################################################

# Proyecto Final - Inove Escuela de Código

# Tema Elegido: Python Analytics

# Alumno: Valentín Imperio.

# Versión: 1.0

# Profesores: Johana Rangel y Julián Salinas.

# Curso: Programador Python.

# Fecha de Entrega: 28/01/2023

# ARCHIVO BONUS: GRAFICOS.PY

#######################################################################

#### Librerías utilizadas ###

import matplotlib.pyplot as plt

import numpy as np

from main import read\_db, ventas\_pais, calzado\_pais, ventas\_genero\_pais

# GRAFICOS DEL PROYECTO #

# Gráfico para Ventas por país

def grafico\_ventas(countries, country, price):

    dict = ventas\_pais(countries, country, price)

    def func(pct, allvals):

        absolute = int(np.round(pct/100.\*np.sum(allvals)))

        return "{:.1f}%\n({:d} $)".format(pct, absolute)

    X = list(dict.values())

    Y = dict.keys()

    # Creo mi paleta de colores:

    colors = ["#7FFFD4","#13EAC9","#04D8B2","#008080"]

    fig = plt.figure()

    fig.suptitle("Ventas de calzado por país", fontsize = 16, weight = "bold")

    ax = fig.add\_subplot()

    ax.pie(X, labels=Y, colors = colors,  autopct=lambda pct: func(pct, X), wedgeprops={'linewidth': 3.0, 'edgecolor': 'white'})

    ax.set\_title("En Dólares [$]", fontsize = 12)

    ax.axis("equal")

    plt.show()

# Gráfico para N° de calzado por país

def grafico\_calzado(countries, country, size):

    dict = calzado\_pais(countries, country, size)

    X = list(dict.keys())

    Y = [float(x) for x in list(dict.values())]

    fig = plt.figure()

    ax = fig.add\_subplot()

    ax.bar(X,Y, color="tab:purple", label="Países")

    ax.set\_title("Tamaño de calzado más vendido en cada país ", fontsize = 10, weight = "bold")

    ax.set\_ylabel(" Tamaño (N° de calzado) ")

    ax.set\_xlabel(" Países ")

    ax.legend()

    plt.xticks(X)

    plt.ylim(top=10)

    plt.show()

# Gráfico para cantidad de ventas por genero elegido para cada país

def grafico\_genero\_pais(countries, gender\_target, country, gender):

    dict = ventas\_genero\_pais(countries, gender\_target, country, gender)

    def func(pct, allvals):

        absolute = int(np.round(pct/100.\*np.sum(allvals)))

        return "{:.1f}%\n({:d} u)".format(pct, absolute)

    X = list(dict.values())

    Y = dict.keys()

    # Creo mi paleta de colores:

    colors = ["#0343DF","#069AF3","#7BC8F6","#000080"]

    fig = plt.figure()

    fig.suptitle("Ventas de calzado para el genero "+gender\_target+" por país", fontsize = 16, weight = "bold")

    ax = fig.add\_subplot()

    \_, \_, autotexts = ax.pie(X, labels=Y, colors = colors,  autopct=lambda pct: func(pct, X), wedgeprops={'linewidth': 3.0, 'edgecolor': 'white'})

    for ins in autotexts:

        ins.set\_color('white')

    ax.set\_title("En pares [u]", fontsize = 12)

    ax.axis("equal")

    plt.show()

####### Bloque principal del programa #######

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

    data = read\_db()

    country = data[0]

    gender = data[1]

    size = data[2]

    price = data[3]

    countries = ["Canada","Germany","United States", "United Kingdom"]

    gender\_target = "Male" # Las opciones a analizar son: "Female", "Male" y "Unix"

    grafico\_ventas(countries, country, price)

    grafico\_calzado(countries, country, size)

    grafico\_genero\_pais(countries, gender\_target, country, gender)

    print("Termine! Que buen programa soy ^^")