

OBJETIVOS

METODOLOGÍA

SQL:

importar las tablas de la base de datos - Lucas

EXPLICAR DUPLICADO DE BASE DE DATOS Lucas

Capturas Lucas

Análisis exploratorio del dataset

SELECT POR TABLAS

AÑO APROX DE LOS DATOS (2000)

idiomas (estoy trabajando en los datos, tengo la cantidad de idiomas hablados, podemos hacer un ranking por cantidad de hablantes)

Python:

Conexión

Consultas

Gráficos

Github - Deb

dificultades

MEMORIA DESCRIPTIVA (lo que nos ocurrió al hacer el proyecto, que dificultades tuvimos y como lo solucionamos)

CONCLUSIONES

MEMORIA PROYECTO

DATOS MUNDIALES



Proyecto Integrador Unicorn Academy

Edición 7.0

Integrantes:

- Gómez Lucas
- Rey Gonzalo
- Vázquez, Débora

Febrero 2025

Unicorn Project | 31/01/2025

1. Contexto y Objetivos del proyecto

2. Herramientas y Software

Para la resolución de la Sección 2, utilizaremos el software Jupyter Notebook de Anaconda Inc.

3. Datos Utilizados (Dataset) (datos ya estaban limpios) (Deby)

4. Ingesta/importación base de datos (llamado al anexo 1) (sección 1) - (Pdf Pocho)

5. Análisis de Datos (análisis exploratorio)

Años del dataset 2000

Valores null

6. Conexión a MySQL desde Python (sección 2) (TERMINAR LAS ACLARACIONES, consulta 6, conclusión idioma vs porcentaje)

6.1. Carga biblio

6.2. funciones

6.3. Consultas y visualizaciones

7. Consideraciones Éticas y de Privacidad 4(Deby)

8. Gestión del Proyecto mediante GitHub (Deby)

9. Conclusiones 26 (Gonza)

Formato (Gonza)

ÍNDICE

1. Contexto y Objetivo del Proyecto	4
2. Herramientas y Software	4
3. Datos Utilizados	4
4. Importación de la Base de Datos: World en MySQL	4
5. Análisis de Datos	5
6. Conexión a MySQL desde Python	5
a. Carga de Bibliotecas	5
b. Definición de Funciones	5
c. Consideración muy importante	7
d. Consultas y visualizaciones	7
7. Consideraciones Éticas y de Privacidad	21
8. Gestión del Proyecto mediante GitHub	21
9. Conclusiones	22

1. Contexto y Objetivo del Proyecto

En un mundo globalizado, la toma de decisiones basadas en datos es esencial para comprender las dinámicas entre países y regiones. Sin embargo, el acceso a información confiable y organizada sigue siendo un desafío debido a la dispersión de los datos en múltiples fuentes y la falta de herramientas eficientes para su análisis.

En este contexto, el proyecto **Datos Mundiales** surge como una solución para consolidar y facilitar la comparación de datos idiomáticos, demográficos, geográficos, económicos y políticos. Su objetivo principal es proporcionar una plataforma centralizada que permita extraer, analizar y comparar esta información de forma rápida, accesible y precisa, ayudando a investigadores, responsables de políticas y otros usuarios a tomar decisiones informadas y a realizar estudios comparativos de alto impacto.

2. Herramientas y Software

El proyecto utiliza las siguientes herramientas y bibliotecas:

- MySQL Workbench: Para la gestión, diseño y administración de bases de datos MySQL.
- Python: Para la manipulación y análisis de datos.
- Jupyter Notebook: Para el desarrollo y previsualización de análisis.
- Librerías de Python: `mysql.connector`, `pandas`, `matplotlib`, empleadas para el procesamiento de datos, generación de gráficos.
- Github: Almacenar el código en el repositorio de Github.

3. Datos Utilizados

Para el desarrollo de este proyecto integrador, se utilizaron tres conjuntos de datos fundamentales: “**world_city**”, “**world_county**” y “**world_countrylanguage**”, proporcionados por **Unicorn Academy**. Estos datasets fueron entregados con el proceso de **ETL** (Extracción, Transformación y Carga) previamente realizado, lo que facilitó el trabajo al contar con datos limpios y listos para su análisis.

4. Importación de la Base de Datos en MySQL

Los conjuntos de datos fueron importados a la base de datos ‘proyecto_integrador’ en **MySQL** utilizando los scripts ‘.sql’ propuestos. La importación se llevó a cabo a través del asistente de MySQL Workbench, lo que permitió cargar las tablas correspondientes de manera eficiente.

Para llevar adelante la importación, se pueden seguir los pasos detallados en la Guía Anexa a esta memoria. En la misma, también se explica como duplicar la base de datos, con la finalidad de mantener los datos en crudo en una base original y luego trabajar con las transformaciones en otra base, quedando así la primera como copia de respaldo.

5. Análisis Exploratorio inicial de Datos

Las tablas importadas contienen información clave sobre ciudades, países y lenguas. Las mismas fueron exploradas inicialmente en MySQL Workbench para realizar consultas y análisis preliminares que permitieron entender con qué información estábamos trabajando y sacar las primeras conclusiones, a saber:

-

6. Conexión a MySQL desde Python

a. Carga de Bibliotecas

Antes de comenzar a trabajar con Python, importamos las bibliotecas.

- Utilizaremos **mysql.connector** para hacer la conexión desde Python a la base de datos en MySQL.
- Utilizaremos **pandas** para poder trabajar con Data Frames.
- Utilizaremos **matplotlib.pyplot** para graficar los resultados.

```
import mysql.connector
import pandas as pd
import matplotlib.pyplot as plt
from mysql.connector import Error
```

b. Definición de Funciones

En primer lugar, definimos la función “**obtener_datos_sql**” que nos permitirá poder hacer todas las consultas que queramos sólo variando el input de la misma.

Utilizaremos un bloque “**try-except**” por si al ejecutar los comandos ocurre un error, y para ello habremos importado con anterioridad la clase “**Error**” de la biblioteca mysql.connector.

Dentro del bloque de código de la función, para hacer una conexión a una base de datos a través de la biblioteca mysql.connector es necesario conectarse al servidor con las mismas credenciales utilizadas en Workbench de MySQL.

Para verificar que la conexión fue exitosa utilizaremos el comando “**.is_connected()**”, informando si efectivamente se pudo realizar la conexión. Luego utilizamos un objeto cursor que nos permitirá ejecutar consultas en lenguaje SQL y recuperar las salidas de las mismas desde la base de datos seleccionada. El comando “**.execute(query)**” ejecuta la consulta que tiene como argumento, en nuestro caso una variable que iremos cambiando en función de cada consulta. Con el comando cursor.description recuperaremos de la base de datos la

información de las columnas, y con el comando `“fetchall()”` obtendremos todos los resultados.

El resultado, junto con los nombres de las columnas, se volcará en un data frame que será la solución para cada consulta que ejecutemos.

Antes de cerrar la conexión cerraremos el bloque `“try-except”` con un código que nos mostrará un texto indicando que ocurrió un error, indicando el tipo de error ocurrido.

Finalmente, si el código pudo realizar la conexión con la base de datos, se cierra el cursor, se cierra la conexión y lo informa.

```
def obtener_datos_sql(query):
    try:
        conexion = mysql.connector.connect(host="localhost",
        user="USER", password="PASSWORD", database= "proyecto_integrador")

        if conexion.is_connected():
            print("Conexión exitosa a la base de datos")
            cursor = conexion.cursor()
            cursor.execute(query)
            columnas = [i[0] for i in cursor.description]
            resultados = cursor.fetchall()
            df = pd.DataFrame(resultados, columns=columnas)
            return df

        except Error as e:
            print(f"Ocurrió un error: {e}")
            return None

        finally:
            if conexion.is_connected():
                cursor.close()
                conexion.close()
                print("Conexión cerrada")
```

En segundo lugar, para poder actualizar datos dentro de la base de datos, definimos la función `“actualizar_datos_sql”`, cuyo argumento será un código de UPDATE en lenguaje SQL. Inicialmente, se intentó utilizar la función `“obtener_datos_sql”` para actualizar la información en la base de datos. Sin embargo, al ejecutar esta función con una instrucción UPDATE, se produjo un error, ya que este tipo de instrucción no genera un conjunto de resultados de salida. La función `“obtener_datos_sql”` quedó diseñada exclusivamente para recuperar datos, no para modificarlos. Para solucionar este problema, se desarrolló una nueva función que se encarga específicamente de actualizar los registros en la base de datos.

La nueva función tiene la misma lógica que la anterior para la conexión y consulta, sólo que en esta se agrega el comando `“commit()”` para efectuar los cambios en la base de datos.

```
def actualizar_datos_sql(update):
    try:
        conexion = mysql.connector.connect(host="localhost",
        user="USER", password="PASSWORD", database= "proyecto_integrador")

        if conexion.is_connected():
            print("Conexión exitosa a la base de datos")
            cursor = conexion.cursor()
            cursor.execute(update)
            conexion.commit()
            print("Datos actualizados")
    except Error as e:
        print(f"Ocurrió un error: {e}")
        return None

    finally:
        if conexion.is_connected():
            cursor.close()
            conexion.close()
            print("Conexión cerrada")
```

c. Consideración muy importante

Este código posibilita al usuario a hacer modificaciones en la base de datos. Para resguardarse de los cambios, antes de ingresar a la misma , es recomendable hacer un backup para resguardar los datos originales de la base de datos. Para ello se pueden seguir los pasos detallados en la guía anexa detallada en el capítulo “Importación de la Base de Datos: World en MySQL” de esta memoria.

Teniendo definidas las funciones, lo que queda es ir definiendo para las variables de entrada de las mismas, las consultas en lenguaje SQL con las que queramos operar en la base de datos.

d. Consultas y visualizaciones

Ejercicio 1: Escribe una consulta para mostrar el nombre y la población de todos los países del continente europeo

```
consulta_sql1 = "SELECT Name as Pais, population as Poblacion FROM
proyecto_integrador.country WHERE continent = 'Europe';"

data_frame1 = obtener_datos_sql(consulta_sql1)
if data_frame1 is not None:
    print(data_frame1)
```


Conexión exitosa a la base de datos

Conexión cerrada

	Pais	Poblacion
0	Albania	3401200
1	Andorra	78000
2	Austria	8091800
3	Belgium	10239000
4	Bulgaria	8190900
5	Bosnia and Herzegovina	3972000
6	Belarus	10236000
7	Switzerland	7160400
8	Czech Republic	10278100
9	Germany	82164700
10	Denmark	5330000
11	Spain	39441700
12	Estonia	1439200
13	Finland	5171300
14	France	59225700
15	Faroe Islands	43000
16	United Kingdom	59623400
17	Gibraltar	25000
18	Greece	10545700
19	Croatia	4473000
20	Hungary	10043200
21	Ireland	3775100
22	Iceland	279000
23	Italy	57680000
24	Liechtenstein	32300
25	Lithuania	3698500
26	Luxembourg	435700
27	Latvia	2424200
28	Monaco	34000
29	Moldova	4380000
30	Macedonia	2024000
31	Malta	380200
32	Netherlands	15864000
33	Norway	4478500
34	Poland	38653600
35	Portugal	9997600
36	Romania	22455500
37	Russian Federation	146934000
38	Svalbard and Jan Mayen	3200
39	San Marino	27000
40	Slovakia	5398700

41	Slovenia	1987800
42	Sweden	8861400
43	Ukraine	50456000
44	Holy See (Vatican City State)	1000
45	Yugoslavia	10640000

Con la biblioteca **Matplotlib.pyplot** haremos visualizaciones a partir de las salidas de las consultas.

Debido a la dificultad de graficar en una misma escala a todos los países según su población, filtramos el data frame en un Top 10 de países con mayor población en Europa. Luego ordenamos el data frame, y generamos una nueva columna dividiendo “**Poblacion**” por 1.000.000 para poder lograr una mejor representación gráfica.

```
data_frame1_Top10 = data_frame1.nlargest(10, 'Poblacion')
```

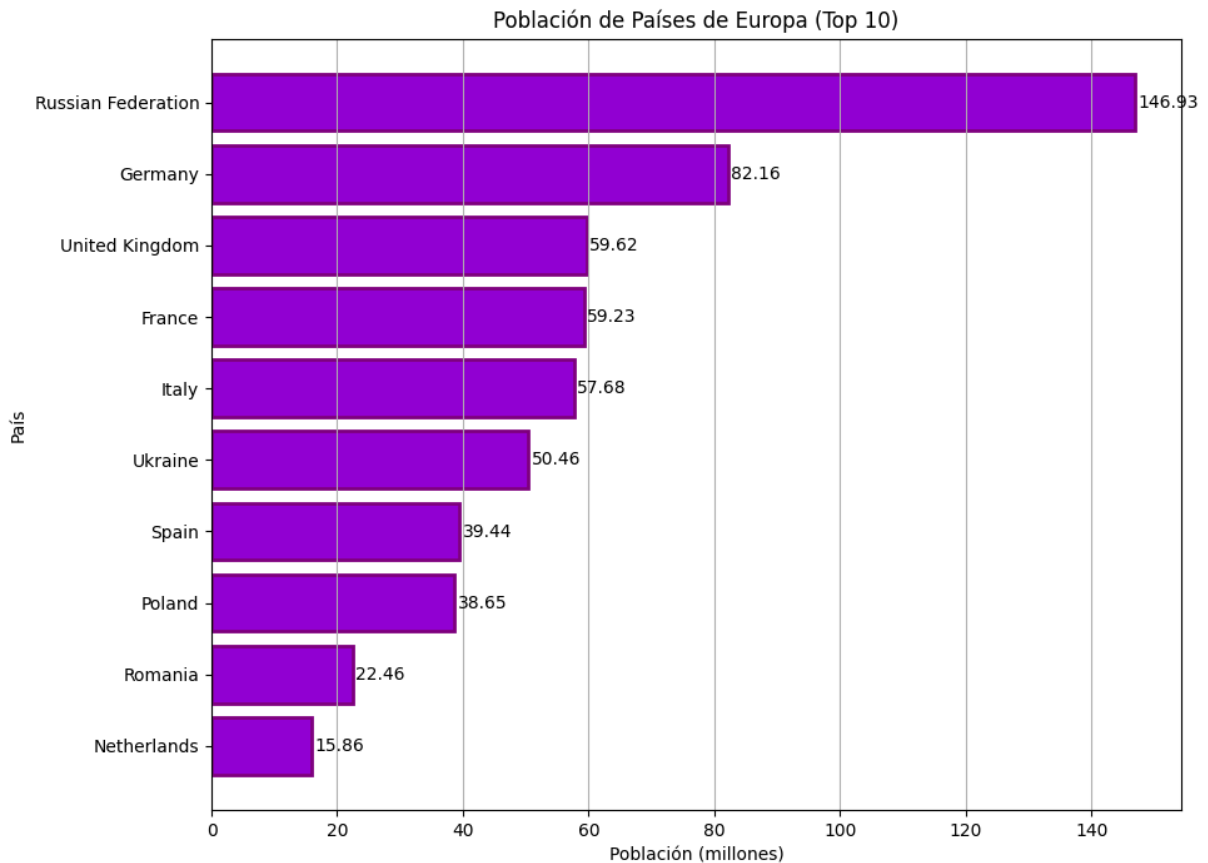
```
data_frame1_Top10 = data_frame1_Top10.sort_values(by='Poblacion',
ascending=True)
```

```
data_frame1_Top10['Poblacion_millones'] =
data_frame1_Top10['Poblacion']/1000000
```

```
plt.figure(figsize=(10, 8))
barras = plt.barh(data_frame1_Top10['País'],
data_frame1_Top10['Poblacion_millones'], color='darkviolet', edgecolor =
"purple", linewidth = 2)
plt.title('Población de Países')
plt.xlabel('Población (millones)')
plt.ylabel('País')

# generamos las etiquetas en las barras con un bucle
for b in barras:
    xval = b.get_width()
    plt.text(xval+0.5, b.get_y() + b.get_height()/2, f"{xval:.2f}",
ha='left', va='center')

plt.grid(axis='x')
plt.show()
```



Ejercicio 2: Escribe una consulta para mostrar los nombres y las áreas de superficie de los cinco países más grandes del mundo (en términos de área de superficie).

```
consulta_sql2 = "SELECT Name as Pais, SurfaceArea as Superficie_km2 FROM
country ORDER BY SurfaceArea desc limit 5;"

data_frame2 = obtener_datos_sql(consulta_sql2)
if data_frame2 is not None:
    print(data_frame2)
```

```
Conexión exitosa a la base de datos
Conexión cerrada
```

	Pais	Superficie_km2
0	Russian Federation	17075400.00
1	Antarctica	13120000.00
2	Canada	9970610.00
3	China	9572900.00
4	United States	9363520.00

Creamos una nueva columna dividiendo “**Superficie_km2**” por 1.000.000 para representar mejor la escala del gráfico.

```
data_frame2['Superficie_millones_km2'] =
data_frame2['Superficie_km2']/1000000
```

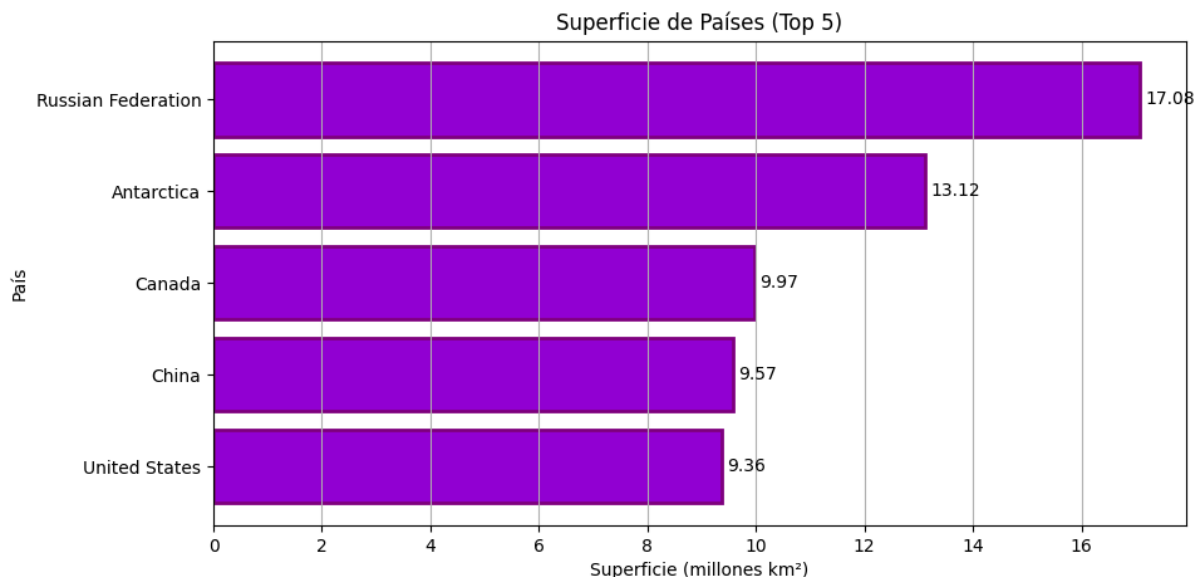
Ordenamos para luego graficar de mayor a menor el Top 5 de países según superficie.

```
data_frame2 = data_frame2.sort_values(by='Superficie_millones_km2',
ascending=True)
```

```
plt.figure(figsize=(10, 5))
barras = plt.barh(data_frame2['País'],
data_frame2['Superficie_millones_km2'], color='darkviolet',
edgecolor="purple", linewidth=2)
plt.title('Superficie de Países (Top 5)')
plt.xlabel('País')
plt.ylabel('Superficie (millones km²)')

# generamos las etiquetas en las barras con un bucle
for b in barras:
    xval = b.get_width()
    plt.text(xval +0.1, b.get_y() + b.get_height()/2, f"{xval:.2f}",
ha='left', va='center')

plt.grid(axis='x')
plt.show()
```



Ejercicio 3: Escribe una consulta para calcular la población total de todos los países de cada continente y mostrar el resultado junto con el nombre del continente.

```

consulta_sql3 = "SELECT Continent as Continente,sum(population) as
PoblacionTotal FROM country GROUP BY continent ORDER BY sum(population)
DESC;"

data_frame3 = obtener_datos_sql(consulta_sql3)
if data_frame3 is not None:
    print(data_frame3)

```

Conexión exitosa a la base de datos
Conexión cerrada

	Continente	PoblacionTotal
0	Asia	3705025700
1	Africa	784475000
2	Europe	730074600
3	North America	482993000
4	South America	345780000
5	Oceania	30401150
6	Antarctica	0

Creamos una nueva columna dividiendo “**PoblacionTotal**” por 1.000.000 para representar mejor la escala del gráfico.

```

data_frame3['PoblacionTotal_Millones'] = data_frame3['PoblacionTotal'] /
1000000

```

```

data_frame3 = data_frame3.sort_values(by='PoblacionTotal',
ascending=True)

```

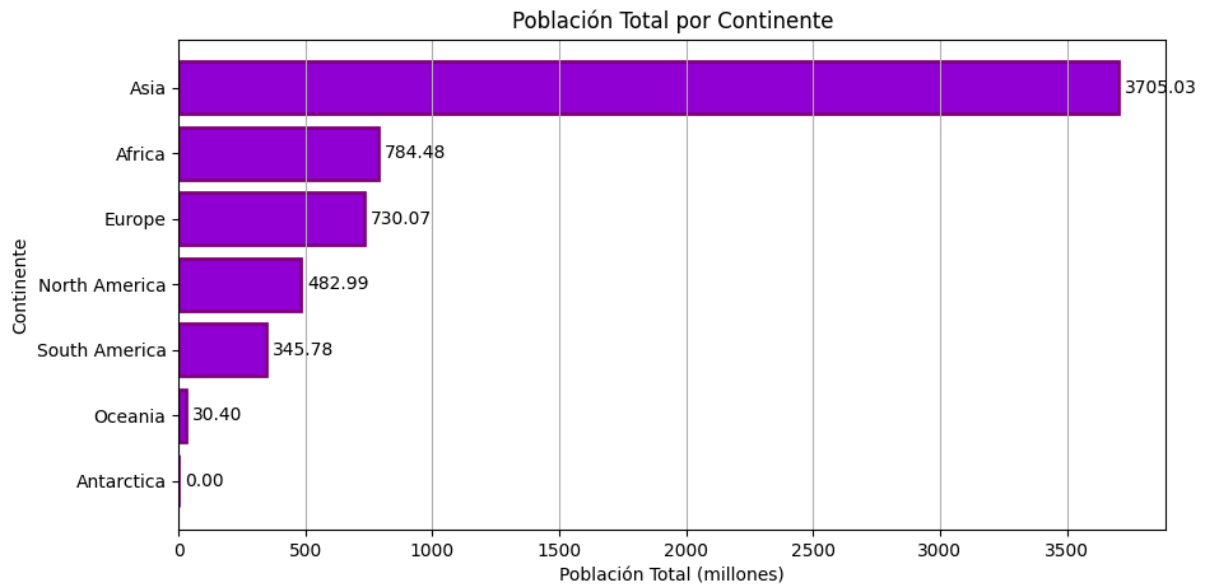
```

plt.figure(figsize=(10, 5))
barras = plt.barh(data_frame3['Continente'],
data_frame3['PoblacionTotal_Millones'], color='darkviolet', edgecolor =
"purple", linewidth = 2)
plt.title('Población Total por Continente')
plt.xlabel('Continente')
plt.ylabel('Población Total (millones)')

# etiquetas en las barras
for b in barras:
    xval = b.get_width()
    plt.text(xval + 25, b.get_y() + b.get_height()/2, f"{xval:.2f}",
ha='left', va='center')

plt.grid(axis='x')
plt.show()

```



Ejercicio 4: Escribe una consulta para mostrar el nombre de las ciudades y la población de todos los países de Europa, ordenados por población de la ciudad de manera descendente.

```
consulta_sql4 = "SELECT ci.name as Ciudad, ci.population as Poblacion,
co.name as Pais FROM city as ci LEFT JOIN country as co ON
ci.countrycode = co.code WHERE co.continent = 'Europe' ORDER BY
ci.POPULATION desc"
```

```
data_frame4 = obtener_datos_sql(consulta_sql4)
if data_frame4 is not None:
    print(data_frame4)
```

Conexión exitosa a la base de datos
Conexión cerrada

	Ciudad	Poblacion	Pais
0	Moscow	8389200	Russian Federation
1	London	7285000	United Kingdom
2	St Petersburg	4694000	Russian Federation
3	Berlin	3386667	Germany
4	Madrid	2879052	Spain
..
836	Serravalle	4802	San Marino
837	San Marino	2294	San Marino
838	Longyearbyen	1438	Svalbard and Jan Mayen
839	Monaco-Ville	1234	Monaco
840	Città del Vaticano	455	Holy See (Vatican City State)

[841 rows x 3 columns]

Para hacer un gráfico primero hicimos una nueva consulta filtrando un Top 20 de ciudades más pobladas de Europa. Luego ordenamos y creamos una nueva columna dividiendo “**Poblacion**” por 1.000.000 para representar mejor la escala del gráfico.

```
consulta_sql4_top = "SELECT ci.name as Ciudad, ci.population as
Poblacion, co.name as Pais FROM city as ci LEFT JOIN country as co ON
ci.countrycode = co.code WHERE co.continent = 'Europe' ORDER BY
ci.POPULATION desc LIMIT 20"
```

```
data_frame4_top = obtener_datos_sql(consulta_sql4_top)
if data_frame4_top is not None:
    print(data_frame4_top)
```

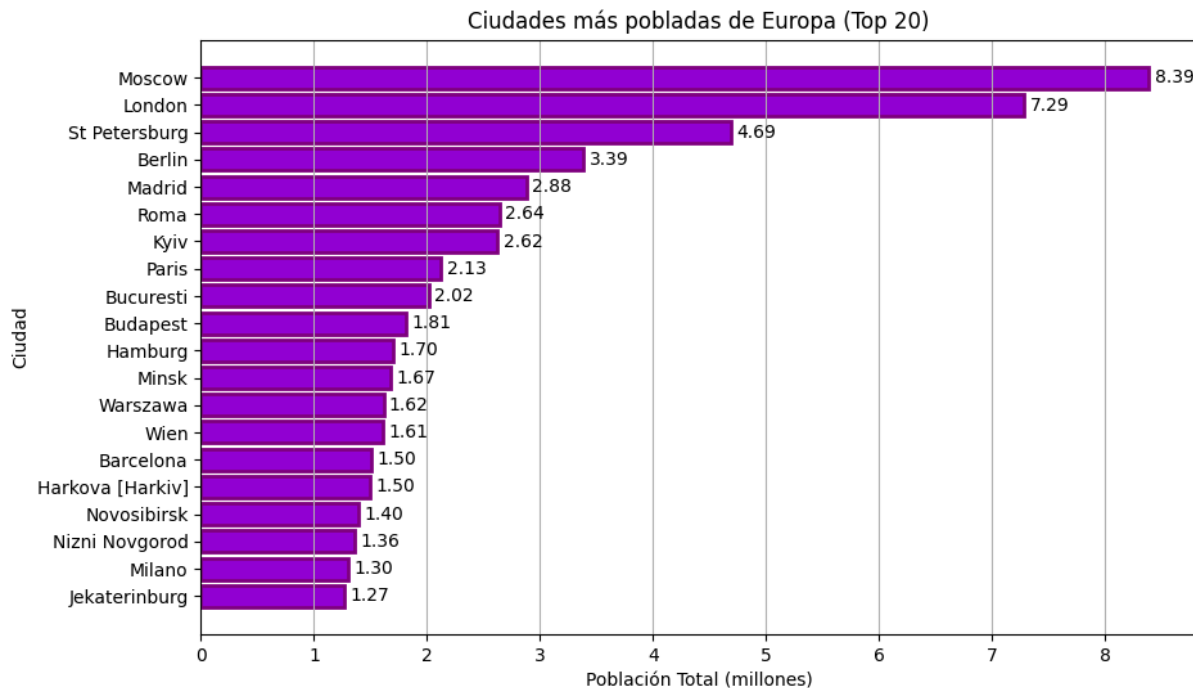
```
data_frame4_top['Poblacion_millones']=data_frame4_top['Poblacion']/1000000
```

```
data_frame4_top = data_frame4_top.sort_values(by='Poblacion_millones',
ascending=True)
```

```
plt.figure(figsize=(10, 6))
barras = plt.barh(data_frame4_top['Ciudad'],
data_frame4_top['Poblacion_millones'], color='darkviolet',
edgecolor="purple", linewidth=2)
plt.title('Ciudades más pobladas de Europa (Top 20)')
plt.xlabel('Población Total (millones)')
plt.ylabel('Ciudad')

# generamos las etiquetas en las barras con un bucle
for b in barras:
    xval = b.get_width()
    plt.text(xval + 0.05, b.get_y() + b.get_height()/2, f"{xval:.2f}",
ha='left', va='center')

plt.grid(axis='x')
plt.show()
```



Ejercicio 5: Actualiza la población de China (código de país 'CHN') a 1500000000 (1.5 mil millones).

Primero haremos la consulta para mostrar la población de China a través del código de país 'CHN'.

```
consulta_sql5 = "SELECT name as Pais, code asCodigoPais, population as
Poblacion FROM country WHERE code = 'CHN'"

data_frame5 = obtener_datos_sql(consulta_sql5)
if data_frame5 is not None:
    print(data_frame5)
```

```
Conexión exitosa a la base de datos
Conexión cerrada
   Pais CodigoPais  Poblacion
0  China         CHN  1277558000
```

Una vez verificada la población, utilizaremos la función “actualizar_datos_sql” para modificar el registro de población para China. Luego volveremos a utilizar la función de consulta para verificar la modificación.

```
# (La población inicial de China era de '1277558000')

update = "UPDATE country SET population = 1500000000 WHERE code =
'CHN';"
actualizar_datos_sql(update)
```



```
data_frame5 = obtener_datos_sql(consulta_sql5)
if data_frame5 is not None:
    print(data_frame5)
```

```
Conexión exitosa a la base de datos
Datos actualizados
Conexión cerrada
Conexión exitosa a la base de datos
Conexión cerrada
   Pais CodigoPais  Poblacion
0  China          CHN  1500000000
```

Ejercicio 6: Cantidad de hablantes por idioma en Europa.

Se incluye esta consulta adicional para obtener la cantidad de hablantes por idioma en el continente europeo. Primero haremos una consulta para unir las tablas “**country**” y “**countrylanguage**” mediante un Join, a través de los campos “**Code**” y “**CountryCode**” respectivamente. Además, partiendo de la columna “**Porcentaje**” multiplicamos por el total de población de cada país y dividimos por 100 para obtener la columna “**Hablantes_Por_Idioma**”, determinando el total de hablantes por idioma por país en el mundo.

```
consulta_sql6 = "SELECT co.Name AS Pais, co.Continent AS Continente,
cl.Language AS Idioma, cl.Percentage AS Porcentaje, ROUND(co.Population
* (cl.Percentage / 100), 0) AS Hablantes_Por_Idioma FROM country co JOIN
countrylanguage cl ON co.Code = cl.CountryCode ORDER BY
ROUND(co.Population * (cl.Percentage / 100), 0) DESC;"

data_frame6 = obtener_datos_sql(consulta_sql6)
if data_frame6 is not None:
    print(data_frame6)
```

	Pais	Continente	Idioma	Porcentaje	Hablantes_Por_Idioma
0	China	Asia	Chinese	92.0	1175353360
1	India	Asia	Hindi	39.9	404451138
2	United States	North America	English	86.2	239943734
3	Brazil	South America	Portuguese	97.5	165862125
4	Russian Federation	Europe	Russian	86.6	127244844
..
979	Saint Vincent and the G	North America	English	0.0	0
980	Virgin Islands, British	North America	English	0.0	0
981	Wallis and Futuna	Oceania	Futuna	0.0	0
982	Wallis and Futuna	Oceania	Wallis	0.0	0
983	Yemen	Asia	Soqutri	0.0	0

Debido a que luego haremos más cálculos, verificamos el tipo de dato de cada columna del data frame obtenido con el método “**.info()**”.

```
data_frame6.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 984 entries, 0 to 983
Data columns (total 5 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Pais                  984 non-null    object
1   Continente            984 non-null    object
2   Idioma                984 non-null    object
3   Porcentaje            984 non-null    object
4   Hablantes_Por_Idioma  984 non-null    object
dtypes: float64(2), object(3)
memory usage: 38.6+ KB
```

Cambiamos los tipos de datos de las columnas numéricas de “**object**” a “**float**” para poder hacer cálculos.

```
data_frame6['Hablantes_Por_Idioma'] =
pd.to_numeric(data_frame6['Hablantes_Por_Idioma'])
data_frame6['Porcentaje'] = pd.to_numeric(data_frame6['Porcentaje'])
```

Filtramos el data frame conservando los países del continente europeo, y los idiomas con “**Porcentaje**” mayores a 0.

```
data_frame6_mod = data_frame6[(data_frame6['Continente'] == 'Europe') &
(data_frame6['Porcentaje'] > 0)]
```

Para calcular el total de población del continente europeo utilizamos una nueva consulta, generando un nuevo data frame con el total de la población por país.

```
consulta_sql_PT = "SELECT Name as Pais, Population as Poblacion FROM
country WHERE Continent = 'Europe' ORDER BY Population DESC;"

Poblacion_Paises_Europa = obtener_datos_sql(consulta_sql_PT)
if Poblacion_Paises_Europa is not None:
    print(Poblacion_Paises_Europa)
```

```
Conexión exitosa a la base de datos
Conexión cerrada

      Pais  Poblacion
0  Russian Federation  146934000
```

```

1          Germany  82164700
2    United Kingdom  59623400
3          France  59225700
4          Italy  57680000
...
41    Liechtenstein  32300
42    San Marino  27000
43    Gibraltar  25000
44    Svalbard and Jan Mayen  3200
45    Holy See (Vatican City State)  1000

[46 rows x 2 columns]

```

A partir del nuevo data frame sumamos el total de población y lo asignamos en la variable **“PobTotalEuropa”**.

```
PobTotalEuropa = Poblacion_Paises_Europa['Poblacion'].sum()
```

Tomamos el data frame inicial, agrupamos los datos según la columna **“Idioma”**, y calculamos el total de hablantes por idioma.

```

Poblacion_Por_Idioma_Europa =
data_frame6_mod.groupby('Idioma')['Hablaantes_Por_Idioma'].sum().reset_in
dex()
print(Poblacion_Por_Idioma_Europa)

```

```

          Idioma  Hablaantes_Por_Idioma
0    Albaniana  5664231.0
1      Arabic  1897294.0
2    Avarian  587736.0
3    Bashkir  1028538.0
4    Basque  631067.0
..
56      Tatar  4701888.0
57    Turkish  3702889.0
58      Udmur  440802.0
59    Ukrainian  35515495.0
60 Ukrainian and Russian  32392.0

[61 rows x 2 columns]

```

Calculamos dos nuevas columnas, una de ellas es **“Hablaantes_Por_Idioma_%”** dividiendo la cantidad de hablantes por idioma por el total de la población europea y multiplicando por 100. La otra columna la obtenemos dividiendo **“Hablaantes_Por_Idioma”** por 1.000.000.

```
Poblacion_Por_Idioma_Europa['Hablaantes_Por_Idioma_%'] =
```

```

round((Poblacion_Por_Idioma_Europa['Hablantes_Por_Idioma']*100 /
PobTotalEuropa),2)
Poblacion_Por_Idioma_Europa['Hablantes_Por_Idioma_millones'] =
round(Poblacion_Por_Idioma_Europa['Hablantes_Por_Idioma']/1000000,2)
print(Poblacion_Por_Idioma_Europa)

```

Finalmente hacemos un Top 10 para cada columna, y reordenamos para representarlas gráficamente.

```

Poblacion_Por_Idioma_Europa = Poblacion_Por_Idioma_Europa.nlargest(10,
'Hablantes_Por_Idioma_%')
Poblacion_Por_Idioma_Europa = Poblacion_Por_Idioma_Europa.nlargest(10,
'Hablantes_Por_Idioma_millones')
Poblacion_Por_Idioma_Europa =
Poblacion_Por_Idioma_Europa.sort_values(by='Hablantes_Por_Idioma_millone
s', ascending=True)

```

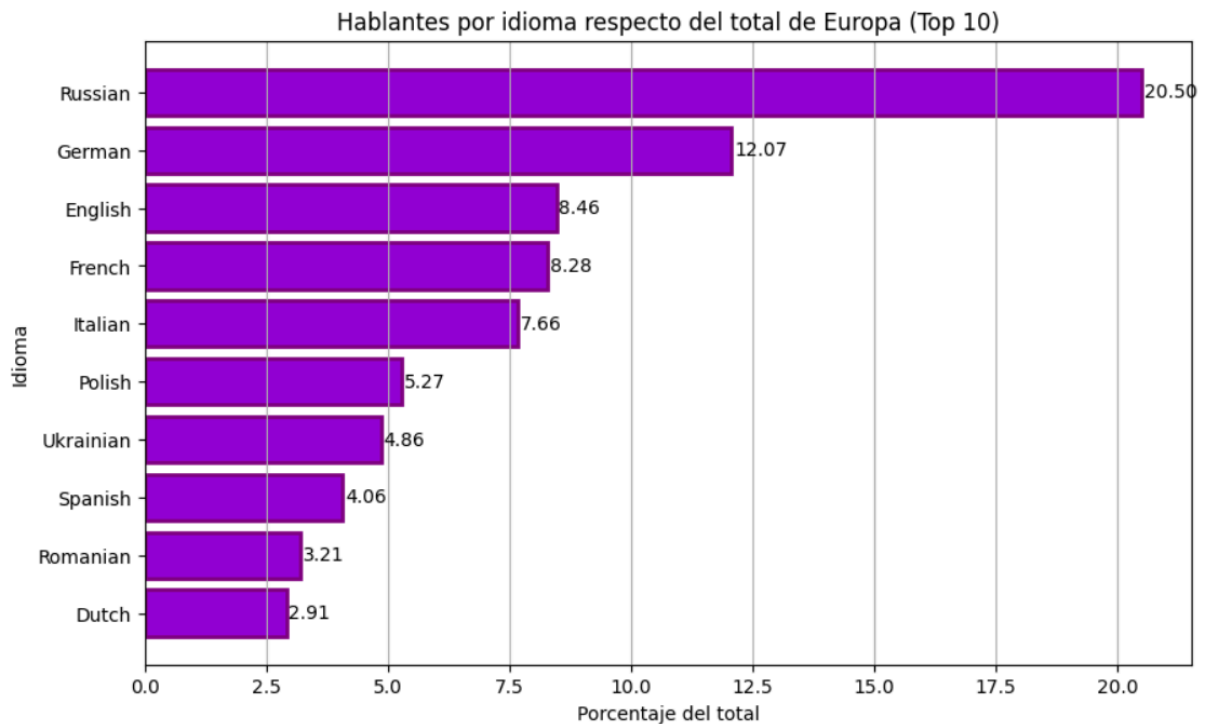
```

plt.figure(figsize=(10, 6))
barras = plt.barh(Poblacion_Por_Idioma_Europa['Idioma'],
Poblacion_Por_Idioma_Europa['Hablantes_Por_Idioma_%'],
color='darkviolet', edgecolor="purple", linewidth=2)
plt.title('Hablantes por idioma respecto del total de Europa (Top 10)')
plt.xlabel('Porcentaje del total')
plt.ylabel('Idioma')

# generamos las etiquetas en las barras con un bucle
for b in barras:
    xval = b.get_width()
    plt.text(xval + 0.05, b.get_y() + b.get_height()/2, f"{xval:.2f}",
ha='left', va='center')

plt.grid(axis='x')
plt.show()

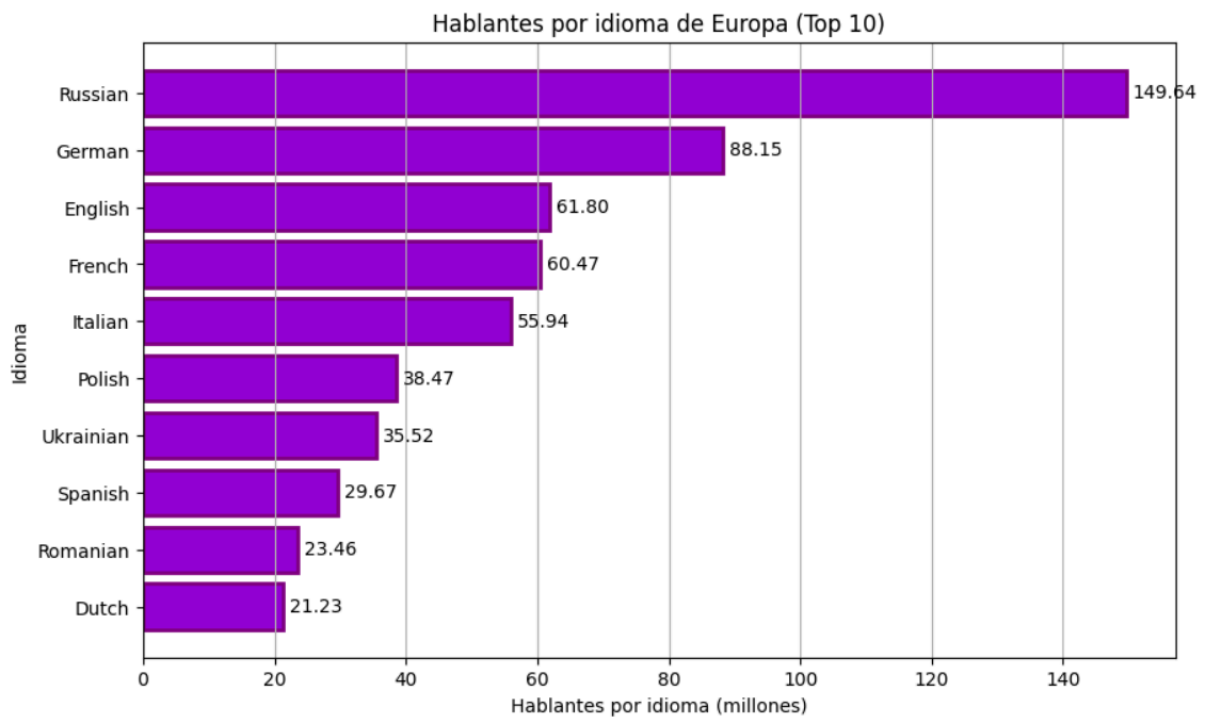
```



```
plt.figure(figsize=(10, 6))
barras = plt.barh(Poblacion_Por_Idioma_Europa['Idioma'],
Poblacion_Por_Idioma_Europa['Hablantes_Por_Idioma_millones'],
color='darkviolet', edgecolor="purple", linewidth=2)
plt.title('Hablantes por idioma de Europa (Top 10)')
plt.xlabel('Hablantes por idioma (millones)')
plt.ylabel('Idioma')

# generamos las etiquetas en las barras con un bucle
for b in barras:
    xval = b.get_width()
    plt.text(xval + 1, b.get_y() + b.get_height()/2, f"{xval:.2f}",
ha='left', va='center')

plt.grid(axis='x')
plt.show()
```



7. Consideraciones Éticas y de Privacidad

Los datos utilizados son públicos y provienen de Unicorn Academy, garantizando el cumplimiento de normativas de privacidad y uso ético. Además, se asegura un manejo responsable, preservando el anonimato y la confidencialidad cuando corresponda.

8. Gestión del Proyecto mediante GitHub

El proyecto Datos Mundiales ha sido gestionado y organizado utilizando **GitHub** como plataforma de almacenamiento y control de versiones. Todo el código desarrollado, incluidas las consultas SQL, scripts de Python y configuraciones necesarias para el análisis de datos, se encuentra almacenado en un repositorio de GitHub. Esto garantiza una estructura ordenada, facilita el acceso y asegura la trazabilidad de los cambios realizados, permitiendo también la colaboración y documentación continua del proyecto.

Repositorio: <https://github.com/Debvaz2024/Proyecto-integrador-bootcamp.git>

9. Conclusiones

En conclusión, el proyecto Datos Mundiales proporciona una solución integral a la dispersión y falta de estandarización de datos globales. Al consolidar información clave de manera eficiente y automatizada, facilita el análisis comparativo, la toma de decisiones informadas y el diseño de estrategias basadas en datos confiables.

El análisis nos ha permitido explorar y analizar datos demográficos y geográficos, con un enfoque particular en Europa, pero también presenta información relevante en términos políticos y económicos, posibilitando hacer otro tipo de análisis y estudios más diversos.

A pesar de ello, este dataset presenta ciertas limitaciones. La más importante de ellas es que los datos no se encuentran actualizados. Identificamos que los datos corresponden alrededor del año 2000, gracias al dato de Fernando de la Rúa como presidente en Argentina.

Algunos puntos interesantes que podemos destacar son los siguientes:

- Europa tiene 46 países dentro de su territorio, y una población total de 730.074.600 habitantes.
- El país con mayor extensión territorial es Rusia, ubicado en el continente europeo.
- Moscú (Rusia), Londres (Reino Unido) y San Petersburgo (Rusia) son las tres ciudades más pobladas de Europa.
- En el mundo existen 458 idiomas y dialectos nativos, de los cuales 61 son hablados en Europa, siendo el ruso (20,50%), el alemán (12,07%) y el inglés (8,46%) los 3 más hablados.

A medida que avanzamos hacia un futuro cada vez más interconectado, es esencial seguir investigando y analizando estos datos para abordar los desafíos globales de manera efectiva.

Perspectivas para el Futuro

A medida que concluimos este análisis, es importante reconocer que los datos presentados son sólo una parte de un panorama mucho más amplio. Futuros estudios podrían explorar la diversidad cultural y lingüística en relación con el desarrollo económico, analizando cómo la diversidad de idiomas hablados en un país puede influir en su crecimiento y cohesión social. Además, sería interesante investigar la correlación entre indicadores económicos como el PBI y la esperanza de vida, considerando cómo el desarrollo económico de un país afecta la salud y el bienestar de las comunidades.

Estas áreas de investigación no solo ampliarían el alcance de nuestro análisis, sino que también proporcionarían información valiosa para académicos, responsables de políticas y organizaciones no gubernamentales. Por otro lado, estos datos también pueden servir de puntapié inicial para el sector privado, donde por ejemplo un análisis de territorios y de densidad poblacional pueden influir para la creación y/o establecimiento de industrias que exploten potencialmente recursos económicos, ofreciendo soluciones a las necesidades de la población a lo largo del mundo.

Según los datos analizados podemos destacar algunos puntos interesantes:

- En el mundo existen 6.078.749.450 habitantes.
- La población de Asia representa aproximadamente el 61% de la población mundial.
- China es el país con la mayor población en el mundo, con 1.277.558.000 habitantes, aproximadamente el 21% del total mundial.
- Los cinco territorios más grandes del planeta (incluyendo la Antártida) concentran casi el 40% de la superficie mundial.