HERRAMIENTAS PARA INTELIGENCIA ARTIFICIAL

UTPL

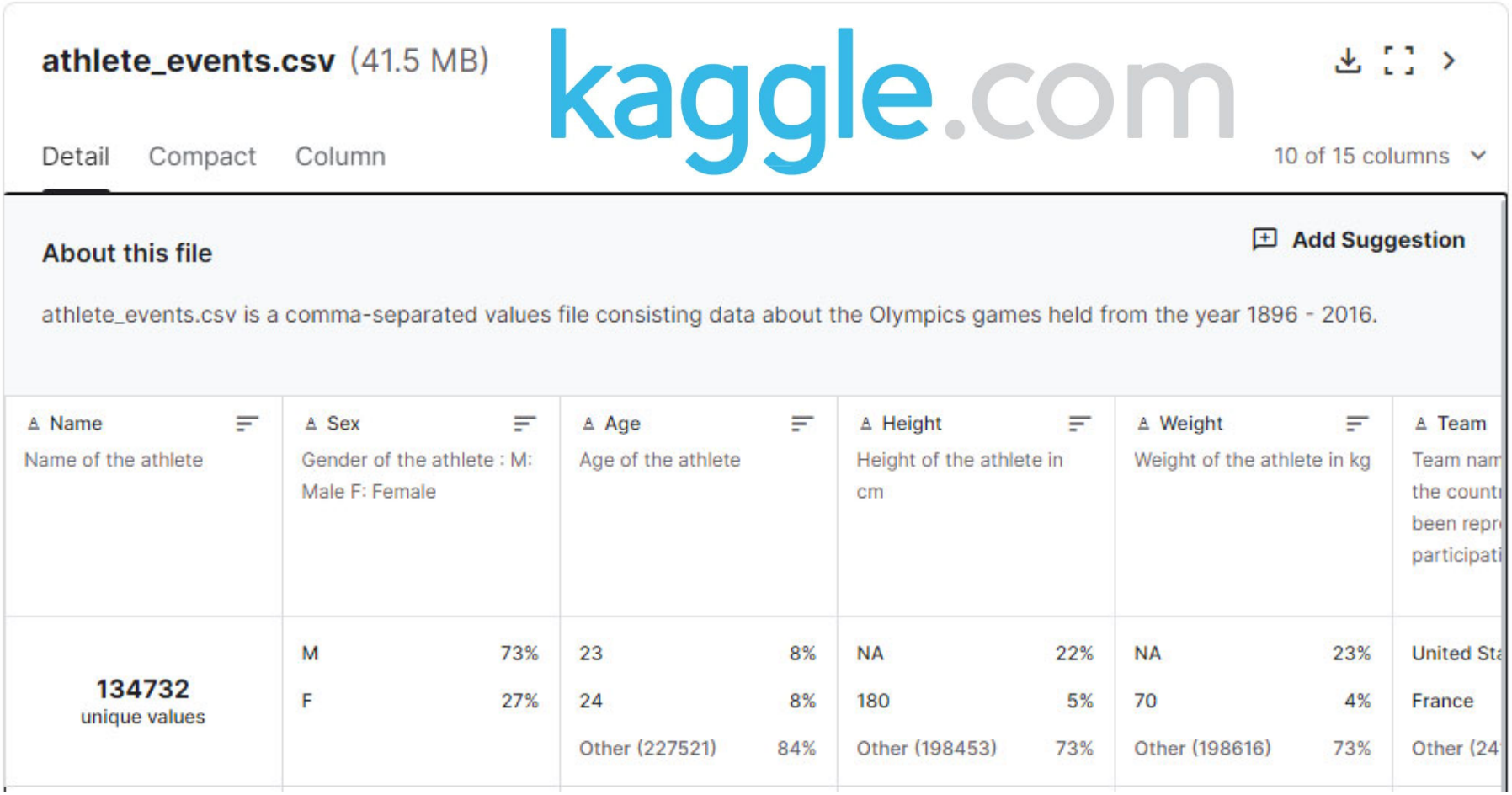
UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA

Noviembre-20

Objetivo

Genera una aplicación de inteligencia artificial que use librerías de software libre a través de herramientas colaborativas.

Origen de datos



athlete_events.csv (41.5 MB)

kaggle.com

Detail Compact Column

10 of 15 columns

About this file

Add Suggestion

athlete_events.csv is a comma-separated values file consisting data about the Olympics games held from the year 1896 - 2016.

Name	Sex	Age	Height	Weight	Team
Name of the athlete	Gender of the athlete : M: Male F: Female	Age of the athlete	Height of the athlete in cm	Weight of the athlete in kg	Team nam the countr been repr participati
134732 unique values	M 73%	23 8%	NA 22%	NA 23%	United Sta
	F 27%	24 8%	180 5%	70 4%	France
		Other (227521) 84%	Other (198453) 73%	Other (198616) 73%	Other (24

https://www.kaggle.com/datasets/samruddhim/olympics-althlete-events-analysis/data

Dimensión del dataset Athletes: (271116, 15)

Origen de datos: Servicio web

POST https://ia.vallext.com/api/v1/login — Send

Params | Authorization | Headers (8) | Body | Scripts | Tests | Settings — Cookies

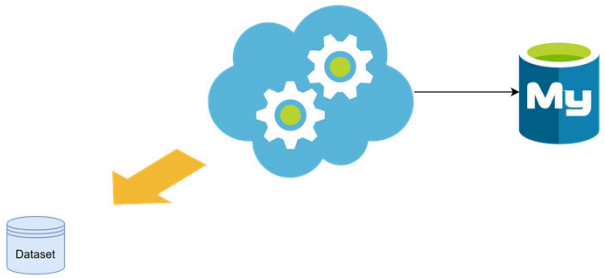
none | form-data | x-www-form-urlencoded | raw | binary | GraphQL

Key		Value	Description	Bulk Edit
email	Text	usuario_api_ia		
password	Text	Pass*2024-IA		
Key	Text	Value	Description	

Body | Cookies | Headers (17) | Test Results — Status: 200 OK | Time: 892 ms | Size: 1.07 KB | Save as example

Pretty | Raw | Preview | Visualize | JSON

```
{
    "success": true,
    "message": "Ok",
    "total": 0,
    "data": {
        "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.
            eyJpc3MiOiJodHRwczovL2lhLnZhbGxleHQuY29tL2FwaS92MS9sb2dpbiIsImlhdCI6MTcxNjQxMzQ5NCwiZXhwIjoxNzE2NDE3MDk0LCJuYmYiOjE3MTY0MTM0OTQsImp0aSI6IkVpV0puN
            zB1VzBCVG5CTkkiLCJzdWIiOiIyIiwicHJ2IjoiMjNiZDVjODk0OWY2MDBhZGIzOWU3MDFjNDAwODcyZGI3YTU5NzZmNyJ9.tamj6NNEr1FIXWvT6sc-dxuqqHd163_gpFy7oDswd5g",
        "token_type": "bearer"
    }
}
```



GET https://ia.vallext.com/api/v1/getTotalAthletesByCountry — Send

Params | Authorization | Headers (7) | Body | Scripts | Tests | Settings — Cookies

Auth Type: Bearer Token — Token: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.ey...

Body | Cookies | Headers (17) | Test Results — Status: 200 OK | Time: 533 ms | Size: 15.76 KB | Save as example

Pretty | Raw | Preview | Visualize | JSON

```
    },
    {
        "code": "AND",
        "name": "Andorra",
        "total_athletes": "61",
        "notes": ""
    },
    {
        "code": "ANG"
```

Origen de datos: Servicio web

```
# Origen 2: Lectura de datos de servicio web
# El servicio utiliza un método de autenticacion mediante JWT
# Inicialmente se debe hacer un login con los datos asignados

# Configuración base
# Url del servicio para autenticación
urlLogin = "https://ia.vallext.com/api/v1/login"

# Url del servicio para obtener los datos
urlServicio = "https://ia.vallext.com/api/v1/getTotalAthletesByCountry"

# Credenciales del servicio (usuario y clave)
payload = {'email': 'usuario_api_ia', 'password': 'Pass*2024-IA'}
files=[]
headers = {}

# Invocación al servicio de autenticación
responseLogin = requests.request("POST", urlLogin, headers=headers, data=payload, files=files)
print(responseLogin.text)
```

```
{"success":true,"message":"Ok","total":0,"data":{"token":"eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwczovL2lhLnZhbGxleHQuY29tL2FwaS92MS9sb2dpbiIsImlhdCI6MTcxNjQwOTEzNSw
IiOiIyIiwicHJ2IjoiMjNiZDVjODk0OWY2MDBhZGIzOWU3MDFjNDAwODcyZGI3YTU5NzZmNyJ9.AqOZPE7JjBSDYLgIZNqNkNOJwJqa9MhQlb3m1sWMvQo","token_type":"bearer"}}
```

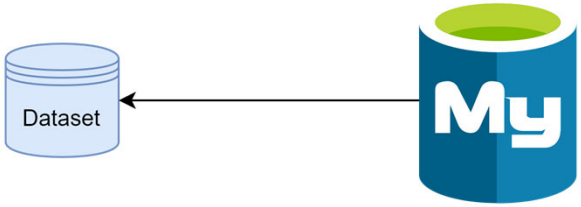
```
# Interpretamos el response del servicio para obtener el JWT
data = json.loads(responseLogin.text)
# Acceder al campo 'token' dentro del diccionario anidado
token = data['data']['token']
print("El token es:", token)
```

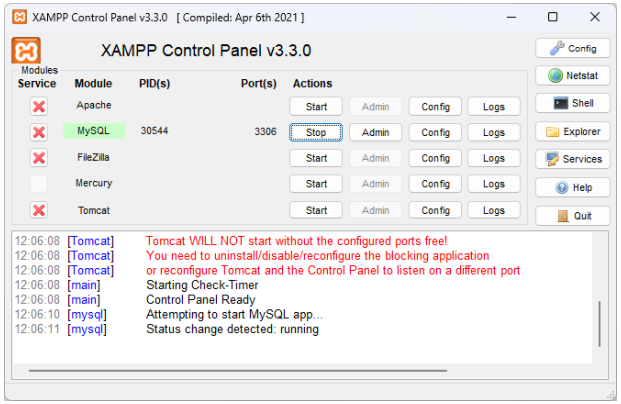
```
El token es: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpc3MiOiJodHRwczovL2lhLnZhbGxleHQuY29tL2FwaS92MS9sb2dpbiIsImlhdCI6MTcxNjQwOTEzNSwiZXhwIjoxNzE2NDEyNzM1LCJuYmYiOjE3MTY0MDkxMzU
3MDFjNDAwODcyZGI3YTU5NzZmNyJ9.AqOZPE7JjBSDYLgIZNqNkNOJwJqa9MhQlb3m1sWMvQo
```

```
# lectura del servicio web para hacer obtener la información adicional
payload = {}
# Configuración del servicio con el método de Authorization - Beare
headers = { 'Authorization': 'Bearer ' + token }
response = requests.request("GET", urlServicio, headers=headers, data=payload)
data = json.loads(response.text)
# Acceder al campo 'data' dentro del json response
dataService = data['data']

# Crear un DataFrame a partir del resultado del servicio
dfCountries = pd.DataFrame(dataService)
dfCountries.sample(20)
```

Origen de datos: Base de datos MySql

	id	codigo	nombre	total_atletas
91470	400	ROU	Romania	1
4522	243	AUS	Australia	1
50449	310	GER	Germany	1
103355	340	KOR	South Korea	1
123288	443	UKR	Ukraine	1
19217	267	CAN	Canada	1
1371	239	ARG	Argentina	1
93344	402	RUS	Russia	1
112552	421	SUI	Switzerland	1
35595	299	FIN	Finland	1



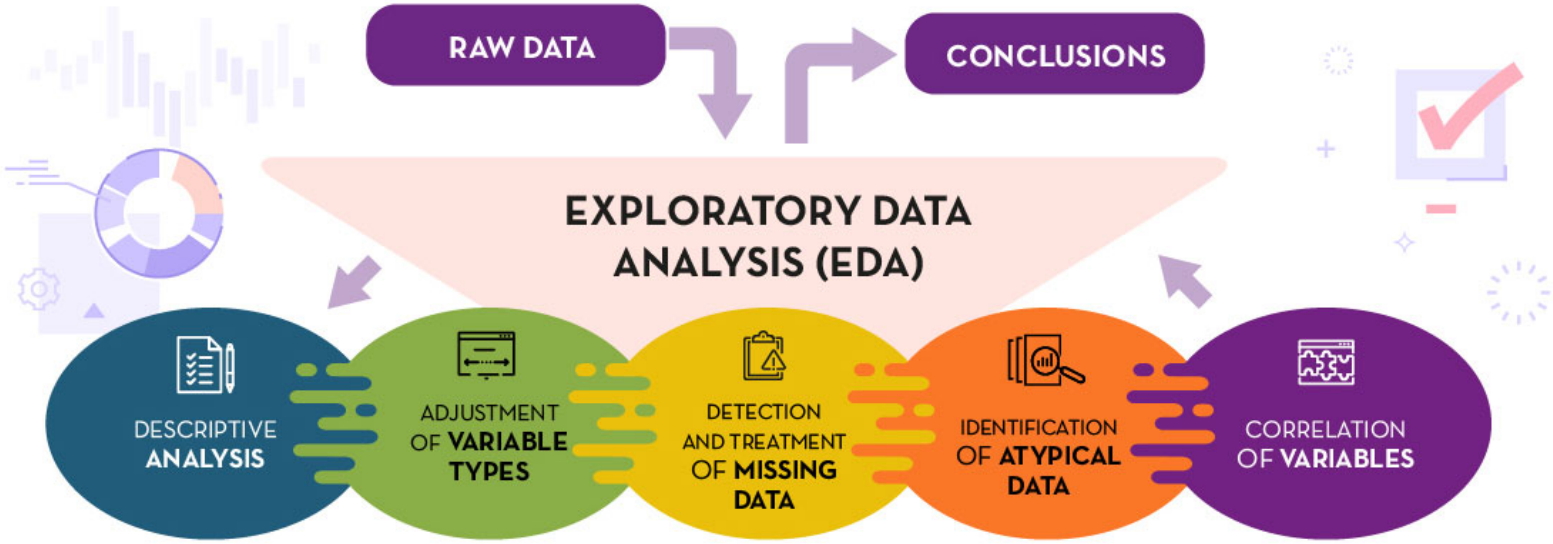



```python
# Origen Opcional
# Método que realiza la conexión a un servidor mysql y ejecuta un query
# Con el resultado creamos un DataFrame
def get_dataframe_from_mysql_server():
    # Parámetros de conexión
    # Estructura del connection string: mysql+pymysql://usuario:contraseña@server/nombre_base_de_datos
    connection_string = 'mysql+pymysql://root:@localhost/db_tools_ia'
    # Query builder
    query = 'SELECT item_catalogo.id, item_catalogo.codigo, item_catalogo.nombre, count(athlete_information.id) AS total_atletas FROM item_catalogo I

    # Creamos el motor de conexión utilizando SQLAlchemy
    engine = create_engine(connection_string)

    # Ejecutamos la consulta y cargamos los resultados en un DataFrame
    with engine.connect() as connection:
        df = pd.read_sql(query, connection)
    return df

dataframe_from_mysql_server = get_dataframe_from_mysql_server()
dataframe_from_mysql_server.sample(10)
```

Análisis exploratorio de datos EDA

El análisis exploratorio de datos (EDA por sus siglas en inglés) implica el uso de gráficos y visualizaciones para explorar y analizar un conjunto de datos. El objetivo es explorar, investigar y aprender, no confirmar hipótesis estadísticas.

Con el EDA, se pueden hallar anomalías en los datos, como valores atípicos u observaciones inusuales, revelar patrones, comprender posibles relaciones entre variables y generar preguntas o hipótesis interesantes que se pueden comprobar más adelante mediante métodos estadísticos más formales.



Análisis exploratorio de datos EDA

```
# Se realiza una revisión inicial del dataframe del servicio y se identifican inconsistencias en la data por lo que inicia un tratamiento EDA

# Función que remueve las tildes de una cadena
def remove_accents(input_str):
    nfkd_form = unicodedata.normalize('NFKD', input_str)
    return "".join([c for c in nfkd_form if not unicodedata.combining(c)])

# Funcion elimina tildes de una columna de una dataframe
def replace_accents_in_column(df, column_name):
    if column_name in df.columns:
        df[column_name] = df[column_name].apply(lambda x: remove_accents(x) if isinstance(x, str) else x)
    else:
        raise ValueError(f"La columna '{column_name}' no existe en el DataFrame.")
    return df

dfCountries = replace_accents_in_column(dfCountries, 'name')
dfCountries.sample(20)
```

```
# Limpieza de caracteres dentro del dataframe
def cleanString(df):
    for column in df.columns:
        if df[column].dtype == 'object':
            df[column] = df[column].apply(lambda x: x.replace('.', '').strip(' ') if isinstance(x, str) else x)
    return df
dfCountries = cleanString(dfCountries)  # Aplicar a cada columna del DF

dfCountries.head(20)
```

Análisis exploratorio de datos EDA

```
# Mostramos la informacion de los 2 DataFrames
print('Dimensión del dataset Athletes: ', athletes.shape, "\n")
athletes.info()
print(60 * '-')
print('Dimensión del dataset Countries: ', dfCountries.shape, "\n")
dfCountries.info()
```

```
Dimensión del dataset Athletes:  (271116, 15)

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271116 entries, 0 to 271115
Data columns (total 15 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   ID      271116 non-null  int64  
 1   Name    271116 non-null  object 
 2   Sex     271116 non-null  object 
 3   Age     261642 non-null  float64
 4   Height  210945 non-null  float64
 5   Weight  208241 non-null  float64
 6   Team    271116 non-null  object 
 7   NOC     271116 non-null  object 
 8   Games   271116 non-null  object 
 9   Year    271116 non-null  int64  
 10  Season  271116 non-null  object 
 11  City    271116 non-null  object 
 12  Sport   271116 non-null  object 
 13  Event   271116 non-null  object 
 14  Medal   39783 non-null   object 
dtypes: float64(3), int64(2), object(10)
memory usage: 31.0+ MB
------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 230 entries, 0 to 229
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   NOC                  230 non-null    object
 1   Region               230 non-null    object
 2   TotalAthletesByNOC   230 non-null    object
 3   Notes                230 non-null    object
 4   married              230 non-null    int32 
 5   country_initial      230 non-null    object
 6   country_athletes     230 non-null    object
 7   reversed_code        230 non-null    object
 8   hash                 230 non-null    object
dtypes: int32(1), object(8)
memory usage: 15.4+ KB
```

Análisis exploratorio de datos EDA

```
# Unificamos los DataFrames
athletes_df=athletes.merge(dfCountries, how='left', on='NOC')
athletes_df.head(15)
```

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	...	Event	Medal	Region	TotalAthletesByNOC	Notes	married	country_initial	country_athletes	reversed_code	hash
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	...	Basketball Men's Basketball	NaN	China	2664		1.0	C	CHN_2664	NHC	AfkNGHCRTX
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	...	Judo Men's Extra-Lightweight	NaN	China	2664		1.0	C	CHN_2664	NHC	AfkNGHCRTX
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	...	Football Men's Football	NaN	Denmark	1922		1.0	D	DEN_1922	NED	y8Vd3L2ocC
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	...	Tug-Of-War Men's Tug-Of-War	Gold	Denmark	1922		1.0	D	DEN_1922	NED	y8Vd3L2ocC
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	...	Speed Skating Women's 500 metres	NaN	Netherlands	2928		1.0	N	NED_2928	DEN	zOE8mMxvDN
5	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	...	Speed Skating Women's 1,000 metres	NaN	Netherlands	2928		1.0	N	NED_2928	DEN	zOE8mMxvDN
6	5	Christine Jacoba Aaftink	F	25.0	185.0	82.0	Netherlands	NED	1992 Winter	1992	...	Speed Skating Women's 500 metres	NaN	Netherlands	2928		1.0	N	NED_2928	DEN	zOE8mMxvDN
7	5	Christine Jacoba Aaftink	F	25.0	185.0	82.0	Netherlands	NED	1992 Winter	1992	...	Speed Skating Women's 1,000 metres	NaN	Netherlands	2928		1.0	N	NED_2928	DEN	zOE8mMxvDN
8	5	Christine Jacoba Aaftink	F	27.0	185.0	82.0	Netherlands	NED	1994 Winter	1994	...	Speed Skating Women's 500 metres	NaN	Netherlands	2928		1.0	N	NED_2928	DEN	zOE8mMxvDN

```
# Mostramos el número de filas y columnas que contiene el DataFrame
athletes_df.shape
```

```
(271116, 23)
```

```
# Mostramos el resumen del DataFrame
athletes_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 271116 entries, 0 to 271115
Data columns (total 23 columns):
 #   Column              Non-Null Count   Dtype
---  ------              --------------   -----
 0   ID                  271116 non-null  int64
 1   Name                271116 non-null  object
 2   Sex                 271116 non-null  object
 3   Age                 261642 non-null  float64
 4   Height              210945 non-null  float64
 5   Weight              208241 non-null  float64
 6   Team                271116 non-null  object
 7   NOC                 271116 non-null  object
 8   Games               271116 non-null  object
 9   Year                271116 non-null  int64
 10  Season              271116 non-null  object
 11  City                271116 non-null  object
 12  Sport               271116 non-null  object
 13  Event               271116 non-null  object
 14  Medal               39783 non-null   object
 15  Region              270767 non-null  object
 16  TotalAthletesByNOC  270767 non-null  object
 17  Notes               270767 non-null  object
 18  married             270767 non-null  float64
 19  country_initial     270767 non-null  object
 20  country_athletes    270767 non-null  object
 21  reversed_code       270767 non-null  object
 22  hash                270767 non-null  object
dtypes: float64(4), int64(2), object(17)
memory usage: 47.6+ MB
```

```
# Obtenemos una visión general estadística rápida de las columnas numéricas en el DataFrame.
# Resumen estadístico que incluye la tendencia central, la dispersión y la forma de la distribución, excluyendo los valores NaN.
# mean,median,statistical values
athletes_df.describe()
```

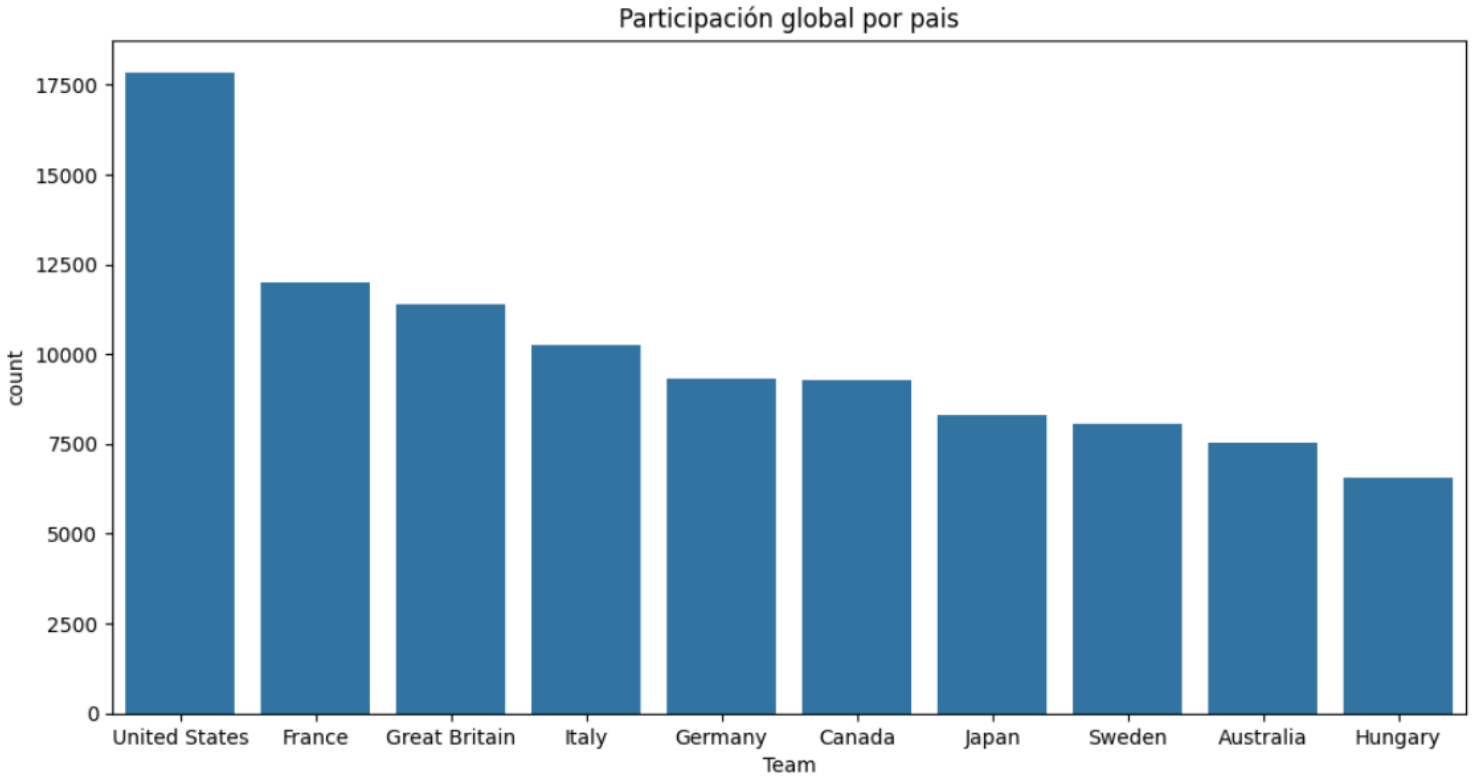
	ID	Age	Height	Weight	Year	married
count	271116.000000	261642.000000	210945.000000	208241.000000	271116.000000	270767.000000
mean	68248.954396	25.556898	175.338970	70.702393	1978.378480	0.528964
std	39022.286345	6.393561	10.518462	14.348020	29.877632	0.499161
min	1.000000	10.000000	127.000000	25.000000	1896.000000	0.000000
25%	34643.000000	21.000000	168.000000	60.000000	1960.000000	0.000000
50%	68205.000000	24.000000	175.000000	70.000000	1988.000000	1.000000
75%	102097.250000	28.000000	183.000000	79.000000	2002.000000	1.000000
max	135571.000000	97.000000	226.000000	214.000000	2016.000000	1.000000

Visualizaciones

```
# Listamos el top de paises participantes
top_10_countries=athletes_df.Team.value_counts().sort_values(ascending=False).head(10)
top_10_countries
```

```
Team
United States    17847
France           11988
Great Britain    11404
Italy            10260
Germany           9326
Canada            9279
Japan             8289
Sweden            8052
Australia         7513
Hungary           6547
Name: count, dtype: int64
```

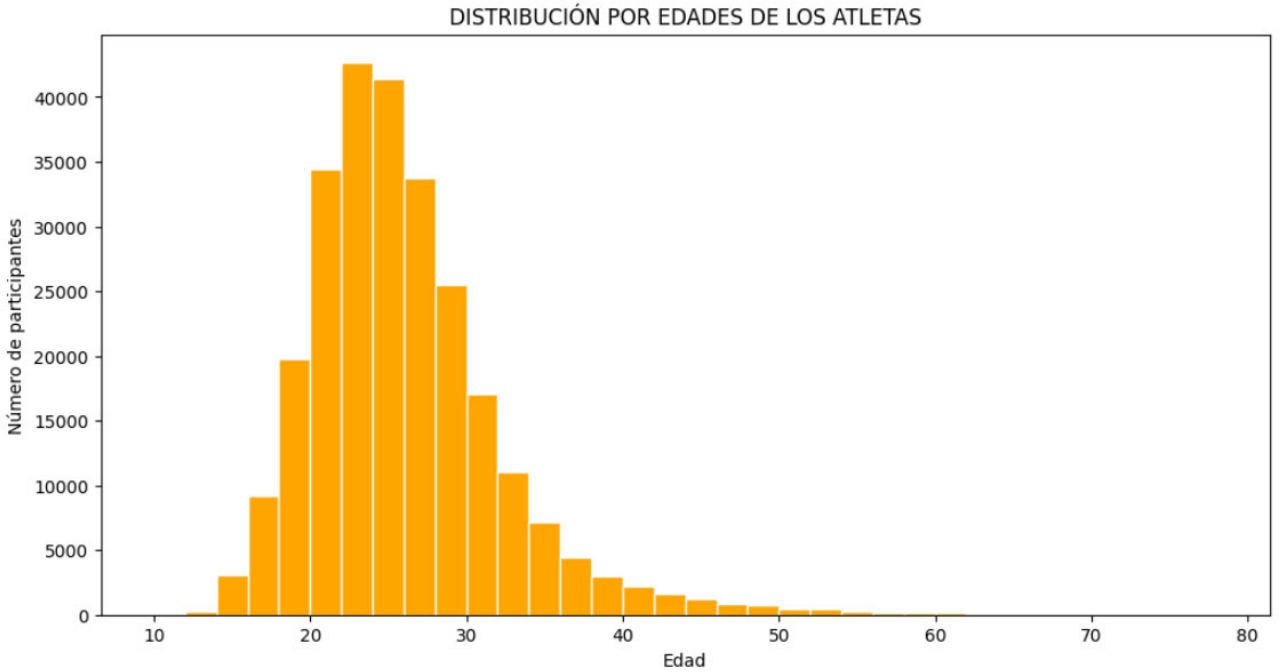
```
# Representación gráfica de la información
# Gráfico de los 10 primeros países
plt.figure(figsize=(12,6))
plt.title('Participación global por pais')
sns.barplot(x=top_10_countries.index, y=top_10_countr
```



Visualizaciones

```
# distribución por edades de los participantes

import seaborn as sns
plt.figure(figsize=(12,6))
plt.title('DISTRIBUCIÓN POR EDADES DE LOS ATLETAS')
plt.xlabel('Edad')
plt.ylabel('Número de participantes')
plt.hist(athletes_df.Age, bins =np.arange(10,80,2),color='orange', edgecolor ='white')
```

```
(array([1.4000e+01, 2.2600e+02, 3.0400e+03, 9.2280e+03, 1.9795e+04,
        3.4422e+04, 4.2689e+04, 4.1427e+04, 3.3700e+04, 2.5506e+04,
        1.7047e+04, 1.1046e+04, 7.1180e+03, 4.4560e+03, 3.0170e+03,
        2.1630e+03, 1.6590e+03, 1.2670e+03, 8.3700e+02, 7.6900e+02,
        4.7700e+02, 4.4400e+02, 2.6600e+02, 2.0000e+02, 1.7100e+02,
        1.5600e+02, 1.1800e+02, 1.1400e+02, 5.6000e+01, 8.5000e+01,
        6.1000e+01, 3.2000e+01, 1.6000e+01, 9.0000e+00]),
 array([10., 12., 14., 16., 18., 20., 22., 24., 26., 28., 30., 32., 34.,
        36., 38., 40., 42., 44., 46., 48., 50., 52., 54., 56., 58., 60.,
        62., 64., 66., 68., 70., 72., 74., 76., 78.]),
 <BarContainer object of 34 artists>)
```

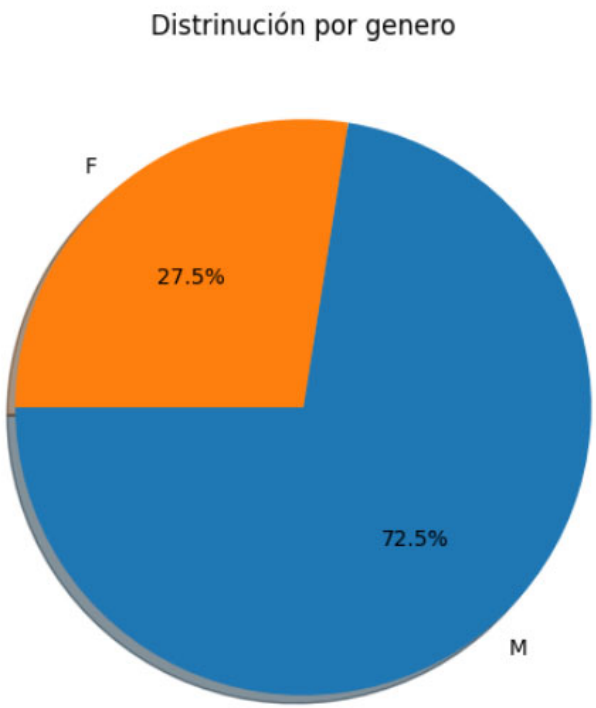


Visualizaciones

```
# Participantes masculinos y femeninos
gender_counts =athletes_df.Sex.value_counts()
gender_counts
```

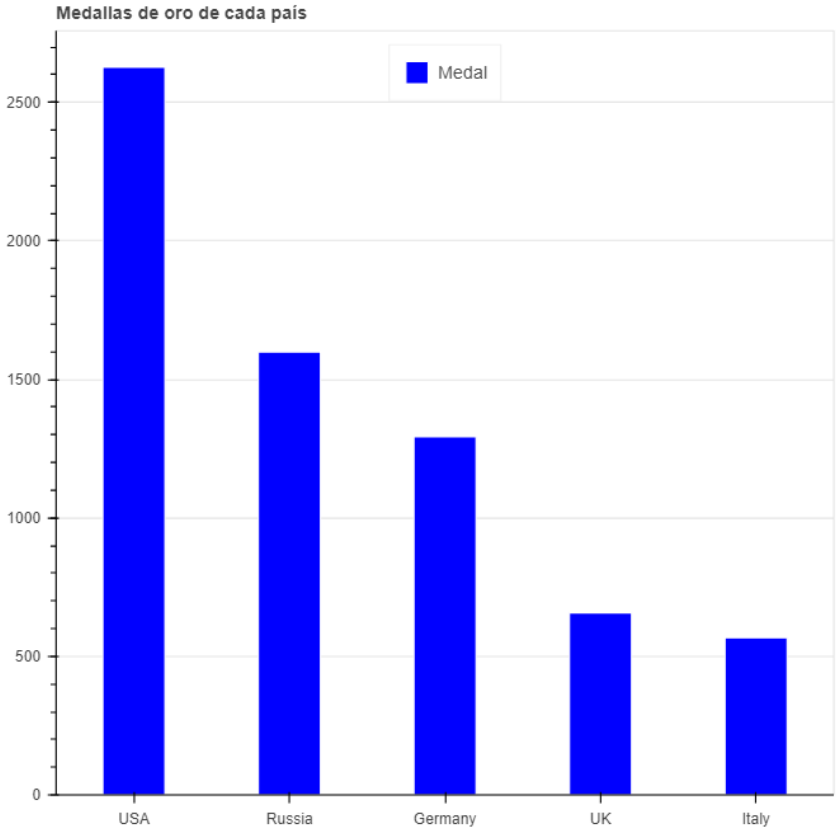
```
Sex
M    196594
F     74522
Name: count, dtype: int64
```

```
# Gráfico circular de atletas masculinos y femeninos
# El parámetro autopct permite mostrar el valor porcentual utilizando el formato de string
plt.figure(figsize=(12,6))
plt.title('Distrinución por genero')
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=180,shadow=True);
```



Visualizaciones Bokeh

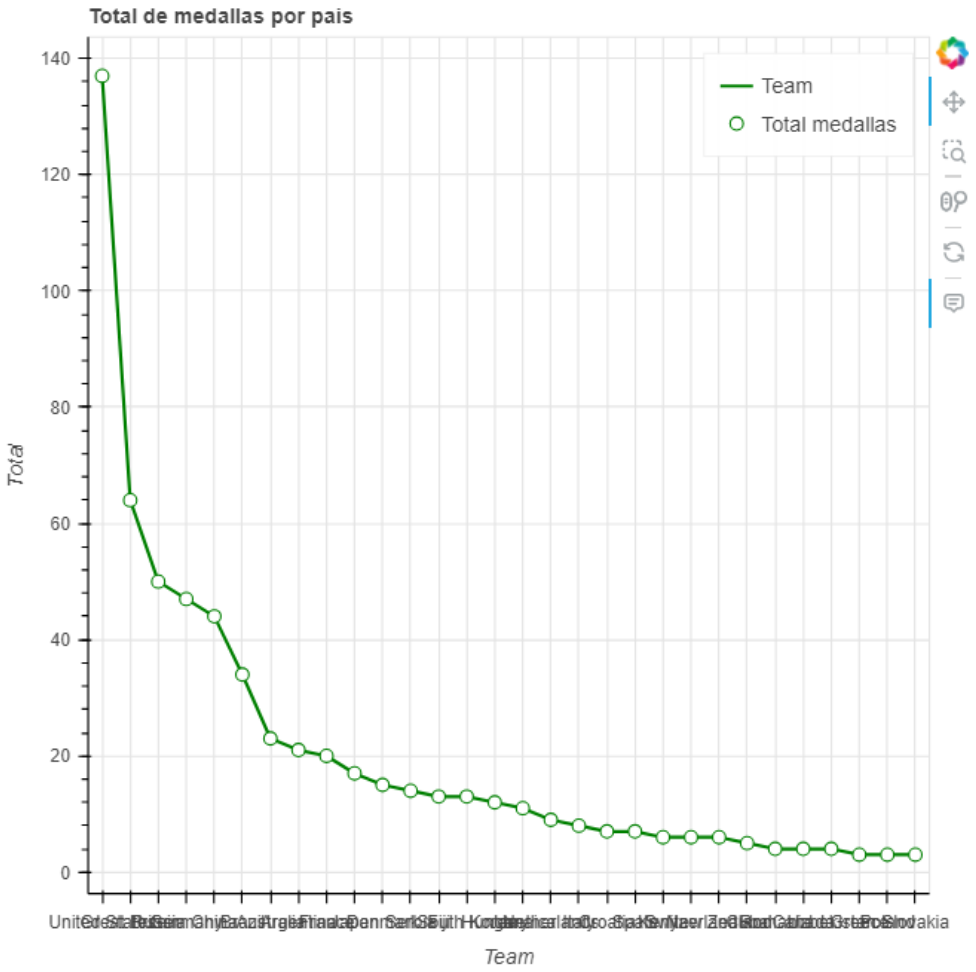
```
p = figure(x_range=dfGoldMedals['Region'].astype(str), title="Medallas de oro de cada país", toolbar_location=None, tools="")

# Añadiendo las barras para los paises con medallas de Oro
p.vbar(x='Region', top='Medal', width=0.4, source=source, legend_label="Medal", color="blue", line_color="white")
p.xgrid.grid_line_color = None
p.y_range.start = 0
p.legend.orientation = "horizontal"
p.legend.location = "top_center"
# Muestra el gráfico
show(p)
```



Visualizaciones Bokeh

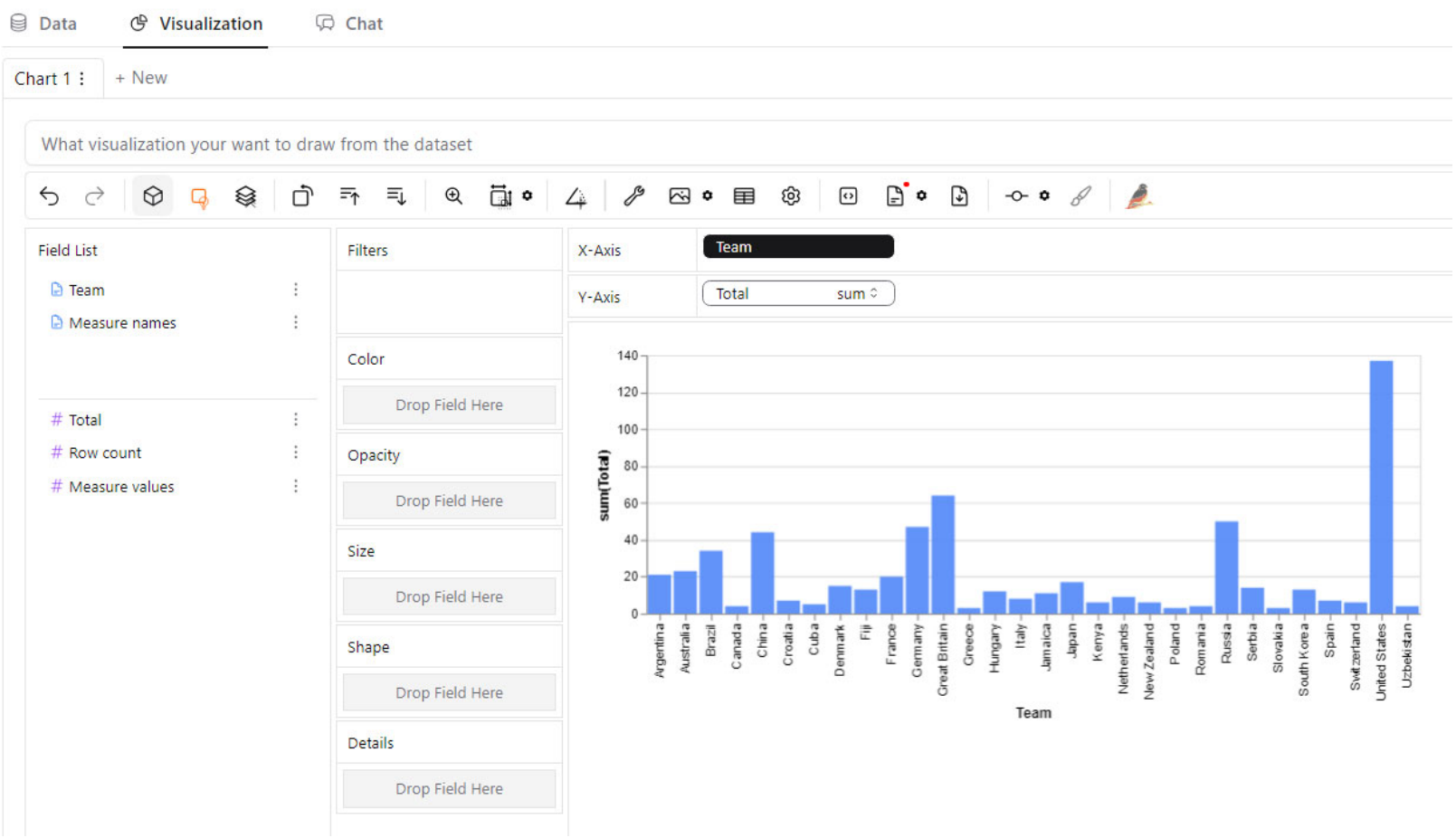
```
# Gráfico 2

gold_medals = athletes_df[(athletes_df.Year == max_year) & (athletes_df.Medal == 'Gold')]

# Contar las medallas de oro por país
gold_medal_count = gold_medals['Team'].value_counts().reset_index().head(30)
gold_medal_count.columns = ['Team', 'Total']
source2 = ColumnDataSource(gold_medal_count)
```

```
# Crear el gráfico
p = figure(title="Total de medallas por pais", x_axis_label='Team', y_axis_label='Total',
           x_range=gold_medal_count['Team'], tools="pan,wheel_zoom,box_zoom,reset")

# Añadir el gráfico de línea
p.line('Team', 'Total', source=source2, line_width=2, color='green', legend_label='Team')
p.scatter('Team', 'Total', source=source2, fill_color="white", size=8, color='green', legend_label='Total medallas')

# Añadir herramienta de hover para mostrar detalles
hover = HoverTool()
hover.tooltips = [
    ("Team", "@Team"),
    ("Total", "@Total")
]
p.add_tools(hover)

# Mostrar el resultado
show(p)
```



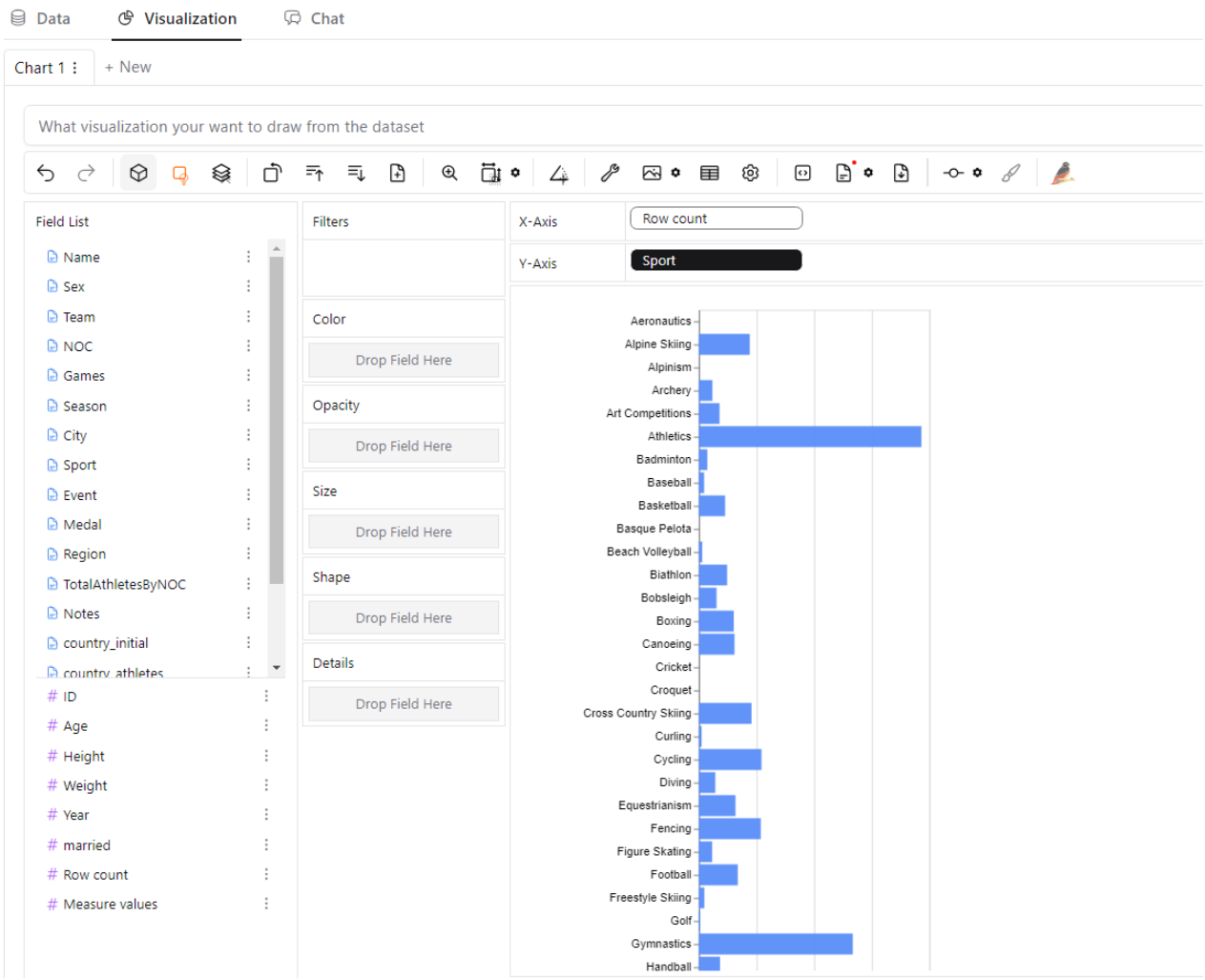
Visualizaciones Pywalker

```
import pygwalker as pyg

# Crear una instancia de Walker con el DataFrame
walker = pyg.walk(gold_medal_count)
```



Visualizaciones Pywalker

```
# Crear una instancia de Walker con otro DatFrame trabajado en este Notebook
walker = pyg.walk(athletes_df)
```



Conclusiones

- El lenguaje de programación Python cuenta con una amplia variedad de bibliotecas especializadas en el análisis de datos, como Pandas, NumPy, SciPy, Matplotlib, Seaborn, Scikit-learn, entre otras. Por lo tanto Python es una excelente opción para el análisis de datos debido a su amplia gama de bibliotecas especializadas.

- Existe varias librerías que permiten realizar graficas de todo tipo en Python, por lo que es recomendable practicar con cada una de ellas, de tal forma que cuando se requiera mostrar información en determinado formato se pueda tener una mejor idea de como y cual de ellas utilizar.

- Python tiene una gran versatilidad al momento de extraer información de diferentes fuentes de datos, como archivos de excel, archivos csv, bases de datos relacionales y no relacionales, así como desde fuentes externas mediante lectura de servicios web en diferentes formatos de datos como es json, xml, etc.

- Es muy importante construir nuestras propias bibliotecas de funciones personalizadas para poder procesar la información en diferentes áreas tales como limpieza de datos, tratamiento de cadenas, detección de valores faltantes, de valores nulos, visualización de datos, etc.

- En lo que tiene que ver con herramientas utilizadas en IA, cada día se actualizan al igual que aparecen otras nuevas con más y más funciones por lo que se requiere entender bien las estructuras de datos donde se pueden aplicar.

Gracias

UTPL

UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA