In [1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
c:\Python312\Lib\site-packages\seaborn\_statistics.py:32: UserWarning: A Num
Py version >=1.22.4 and <2.3.0 is required for this version of SciPy (detect
ed version 2.3.2)
  from scipy.stats import gaussian_kde
```

In [2]:

```python
def read_csv_auto(path: str):
    for enc in ("utf-8", "utf-8-sig", "cp1252", "latin1"):
        try:
            df = pd.read_csv(path, encoding=enc)
            print(f"Loaded {path} with encoding={enc}")
            return df
        except UnicodeDecodeError:
            continue
    # Last resort: ignore undecodable chars
    df = pd.read_csv(path, encoding="latin1", encoding_errors="ignore")
    print(f"Loaded {path} with encoding=latin1 (errors=ignore)")
    return df

# Load datasets with robust encoding handling
stage_data = read_csv_auto('stage_data.csv')
tdf_stages = read_csv_auto('tdf_stages.csv')
tdf_winners = read_csv_auto('tdf_winners.csv')
```

```
Loaded stage_data.csv with encoding=utf-8
Loaded tdf_stages.csv with encoding=utf-8
Loaded tdf_winners.csv with encoding=utf-8
```

Análisis Exploratorio de Datos (EDA)

En esta sección se realiza un EDA de los datasets cargados (`stage_data` , `tdf_stages` , `tdf_winners`).

In [3]:

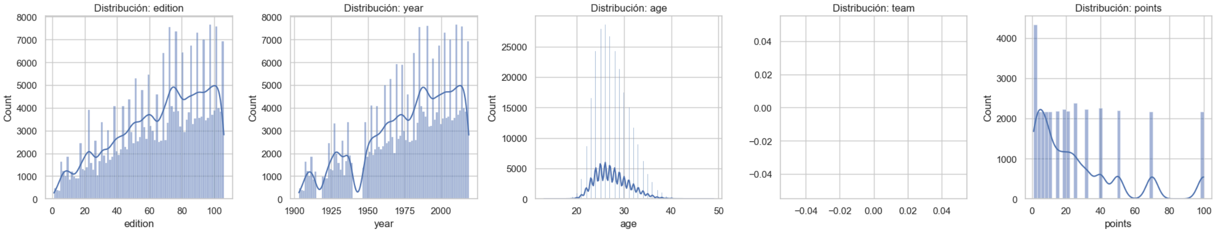
```python
from typing import Optional, List

def quick_eda(df: pd.DataFrame, name: str,
              max_num_cols: int = 6,
              max_cat_cols: int = 6,
              top_n_categories: int = 10,
              corr_max_cols: int = 20,
              figsize=(12, 4)):
    """Genera un EDA compacto para el DataFrame dado y dibuja visuales básic

    Args:
        df: DataFrame a analizar
        name: Nombre lógico del dataset
        max_num_cols: Máximo de columnas numéricas a graficar (histogramas)
        max_cat_cols: Máximo de columnas categóricas a graficar (barras)
        top_n_categories: Top N categorías a mostrar en barras
        corr_max_cols: Máximo de columnas numéricas en el mapa de correlació
        figsize: Tamaño base de las figuras
    """
    import warnings
    warnings.filterwarnings("ignore")

    print(f"\n===== {name} =====")
    print(f"Forma (filas, columnas): {df.shape}")
    print("\nTipos de datos:\n", df.dtypes)

    # Valores faltantes
    na_counts = df.isna().sum()
    na_pct = (na_counts / len(df) * 100).round(2)
    if na_counts.sum() > 0:
        na_table = pd.DataFrame({"na_count": na_counts, "na_pct": na_pct})
        print("\nValores faltantes (no-cero):\n", na_table[na_table.na_count
    else:
        print("\nValores faltantes: ninguno")

    # Duplicados
    dup_count = df.duplicated().sum()
    print(f"\nFilas duplicadas: {dup_count}")

    # Describe
    num_df = df.select_dtypes(include=np.number)
    if num_df.shape[1] > 0:
        desc = num_df.describe().T
        print("\nResumen numérico (describe):\n", desc)
        # Sesgo/curtosis
        sk = num_df.skew(numeric_only=True)
        ku = num_df.kurtosis(numeric_only=True)
        sk_ku = pd.concat([sk.rename("sesgo"), ku.rename("curtosis")], axis=
        print("\nSesgo/curtosis:\n", sk_ku)
    else:
        print("\nNo se detectaron columnas numéricas.")

    # Categóricas
    cat_cols = df.select_dtypes(include=["object", "category"]).columns.toli
    if cat_cols:
        print("\nColumnas categóricas y cardinalidad:")
        for c in cat_cols:
            nunique = df[c].nunique(dropna=True)
            print(f"  - {c}: {nunique} únicas")
    else:
        print("\nNo se detectaron columnas categóricas.")

    # Gráficos
    try:
        sns.set_theme(style="whitegrid")

        num_cols = num_df.columns.tolist()
        if num_cols:
            plot_cols = num_cols[:max_num_cols]
            n = len(plot_cols)
            if n:
                fig, axes = plt.subplots(1, n, figsize=(figsize[0]*n/3, figs
                if n == 1:
                    axes = [axes]
                for ax, col in zip(axes, plot_cols):
                    sns.histplot(df[col].dropna(), kde=True, ax=ax)
                    ax.set_title(f"Distribución: {col}")
                plt.tight_layout()
                plt.show()

        if cat_cols:
            plot_cols = cat_cols[:max_cat_cols]
            n = len(plot_cols)
            if n:
                fig, axes = plt.subplots(1, n, figsize=(figsize[0]*n/3, figs
                if n == 1:
                    axes = [axes]
                for ax, col in zip(axes, plot_cols):
                    vc = df[col].value_counts(dropna=True).head(top_n_catego
                    sns.barplot(x=vc.values, y=vc.index, ax=ax, orient="h")
                    ax.set_title(f"Top {top_n_categories} {col}")
                plt.tight_layout()
                plt.show()

        # Correlación
        if num_cols:
            corr_cols = num_cols[:corr_max_cols]
            corr = df[corr_cols].corr(numeric_only=True)
            plt.figure(figsize=(min(1+0.5*len(corr_cols), 14), min(1+0.5*len
            sns.heatmap(corr, cmap="coolwarm", center=0, annot=False)
            plt.title("Matriz de correlación")
            plt.tight_layout()
            plt.show()

        # Tendencia temporal si hay columna de año/fecha
        def to_year_series(s):
            try:
                return pd.to_numeric(s, errors='coerce')
            except Exception:
                pass
            try:
                return pd.to_datetime(s, errors='coerce').dt.year
            except Exception:
                return pd.Series([np.nan]*len(s), index=s.index)

        year_candidates = [c for c in df.columns if c.lower() in ("year", "a
        if year_candidates and num_cols:
            ycol = year_candidates[0]
            yseries = to_year_series(df[ycol])
            ydf = pd.concat([yseries.rename("__year__"), num_df], axis=1)
            ydf = ydf.dropna(subset=["__year__"])  # Asegurar años válidos
            if not ydf.empty:
                agg = ydf.groupby("__year__")[num_cols].mean(numeric_only=Tr
                ax = agg.plot(figsize=(10,4), title=f"Promedios por año ({yc
                ax.set_xlabel("Año")
                plt.tight_layout()
                plt.show()
```

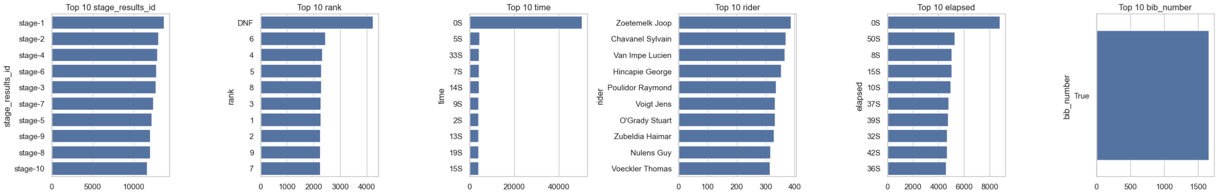
```
    except Exception as e:
        print("Se produjo un error en la sección de gráficos:", e)
```

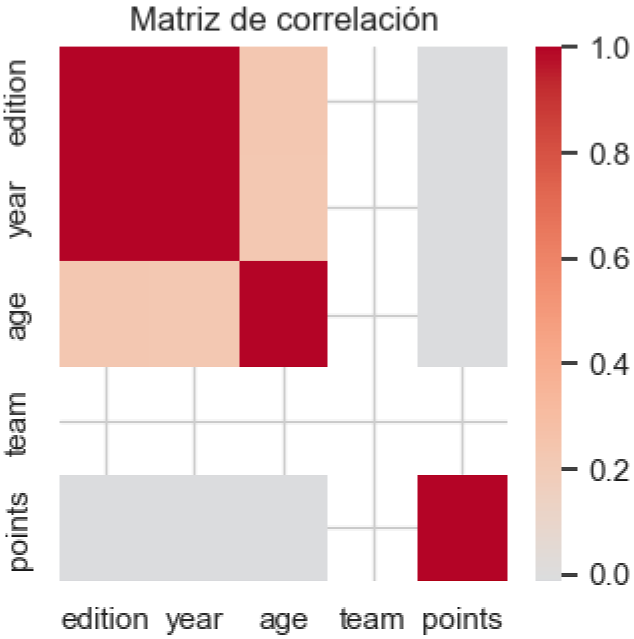
In [4]:

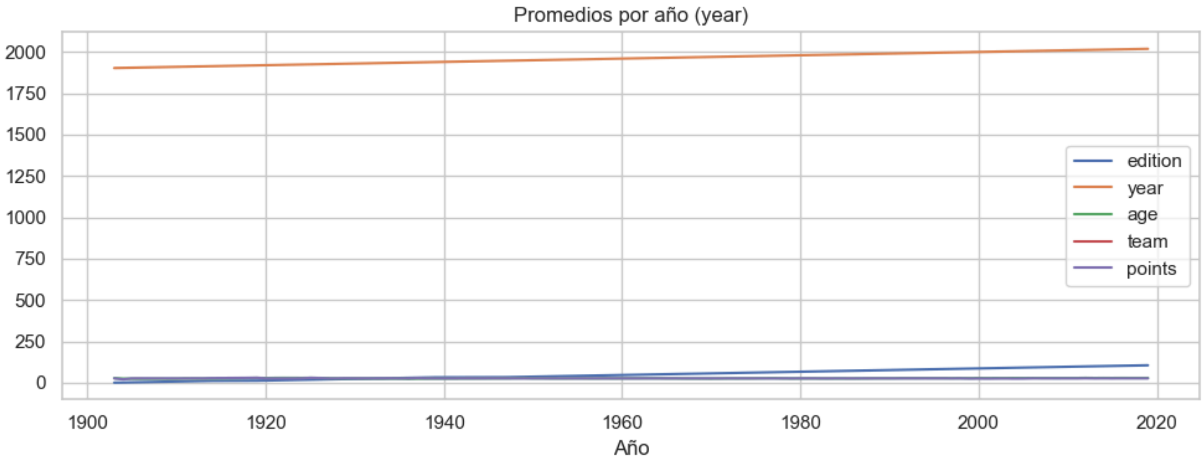
```
# Ejecutar EDA sobre cada dataset

datasets = {
    "stage_data": stage_data,
    "tdf_stages": tdf_stages,
    "tdf_winners": tdf_winners,
}

for name, df in datasets.items():
    if isinstance(df, pd.DataFrame) and not df.empty:
        quick_eda(df, name)
    else:
        print(f"Skipping {name}: not a valid non-empty DataFrame")
```

```
===== stage_data =====
Forma (filas, columnas): (255752, 11)

Tipos de datos:
 edition               int64
year                  int64
stage_results_id     object
rank                 object
time                 object
rider                object
age                 float64
team                float64
points              float64
elapsed              object
bib_number           object
dtype: object

Valores faltantes (no-cero):
             na_count  na_pct
team          255752  100.00
bib_number    254096   99.35
points        222746   87.09
time            5617    2.20
elapsed         5617    2.20
age             3326    1.30

Filas duplicadas: 0

Resumen numérico (describe):
             count         mean        std     min     25%     50%     75%
\
edition  255752.0    66.496950  26.817165     1.0    47.0    71.0    89.0
year     255752.0  1978.342930  28.939706  1903.0  1960.0  1984.0  2002.0
age      252426.0    27.547257   3.636306    13.0    25.0    27.0    30.0
team          0.0          NaN        NaN     NaN     NaN     NaN     NaN
points    33006.0    26.938526  27.035525     1.0     6.0    18.0    40.0

            max
edition   106.0
year     2019.0
age        49.0
team        NaN
points    100.0

Sesgo/curtosis:
             sesgo  curtosis
edition -0.461071 -0.760541
year    -0.646321 -0.433981
age      0.576350  0.178995
team          NaN       NaN
points   1.386451  1.165421

Columnas categóricas y cardinalidad:
  - stage_results_id: 67 únicas
  - rank: 219 únicas
  - time: 60 únicas
- rider: 5162 únicas
- elapsed: 60 únicas
- bib_number: 1 únicas
```

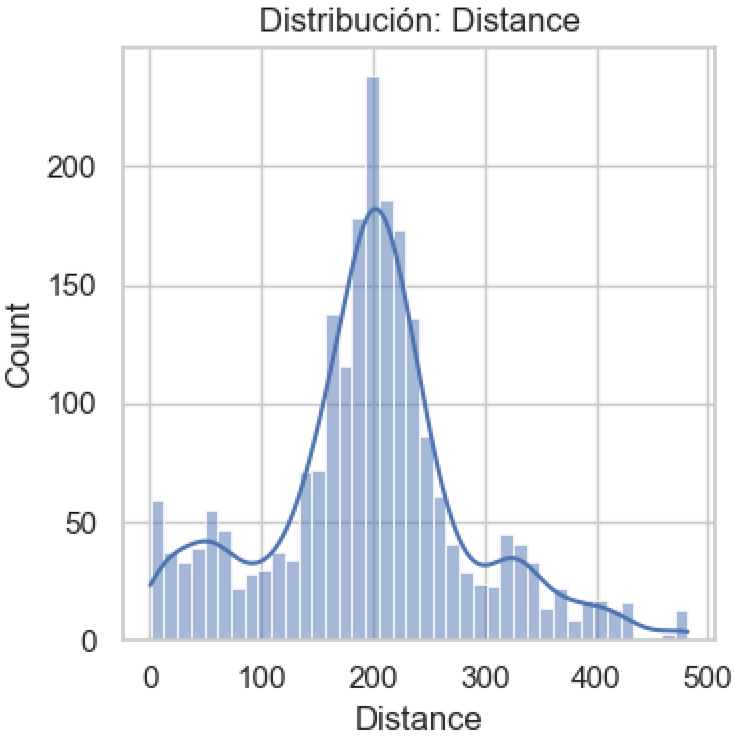


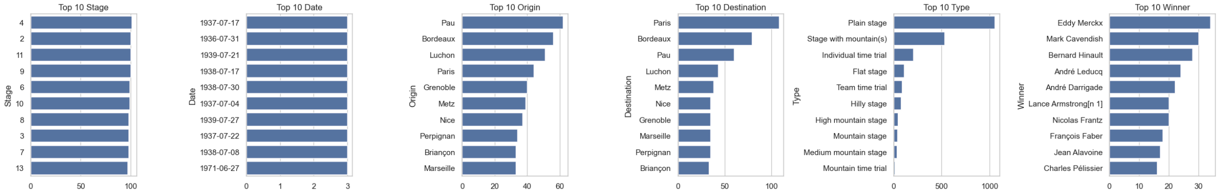


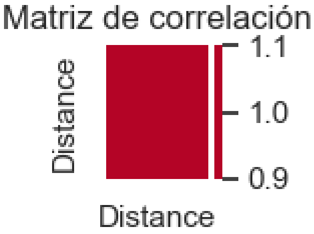




```
===== tdf_stages =====
Forma (filas, columnas): (2236, 8)

Tipos de datos:
 Stage              object
Date               object
Distance          float64
Origin             object
Destination        object
Type               object
Winner             object
Winner_Country     object
dtype: object

Valores faltantes (no-cero):
                 na_count  na_pct
Winner_Country        52    2.33

Filas duplicadas: 0

Resumen numérico (describe):
            count        mean        std  min    25%    50%    75%    max
Distance  2236.0  196.782994  90.176385  1.0  156.0  199.0  236.0  482.0

Sesgo/curtosis:
             sesgo  curtosis
Distance  0.16731  0.524551

Columnas categóricas y cardinalidad:
  - Stage: 80 únicas
  - Date: 2113 únicas
  - Origin: 591 únicas
  - Destination: 514 únicas
  - Type: 18 únicas
  - Winner: 878 únicas
  - Winner_Country: 40 únicas
```

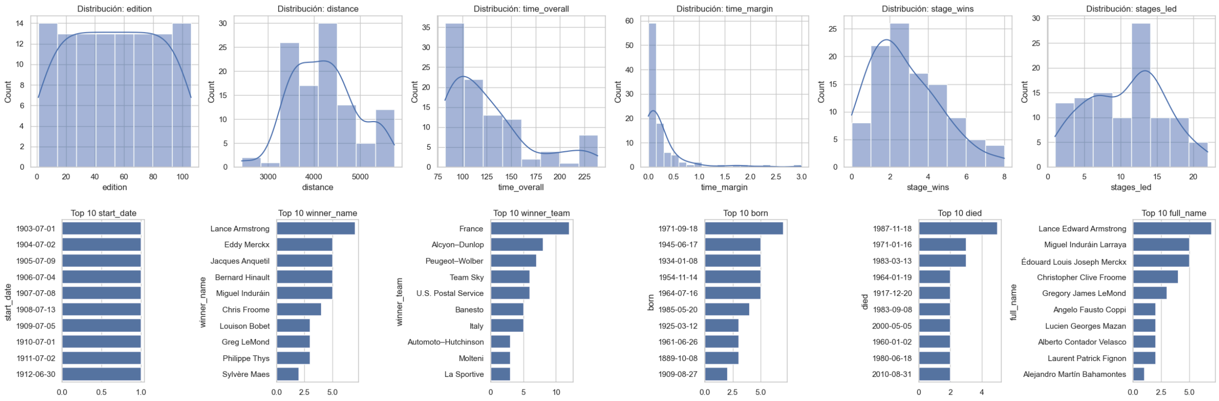

Distribución: Distance
Count
0
50
100
150
200
0
100
200
300
400
500
Distance

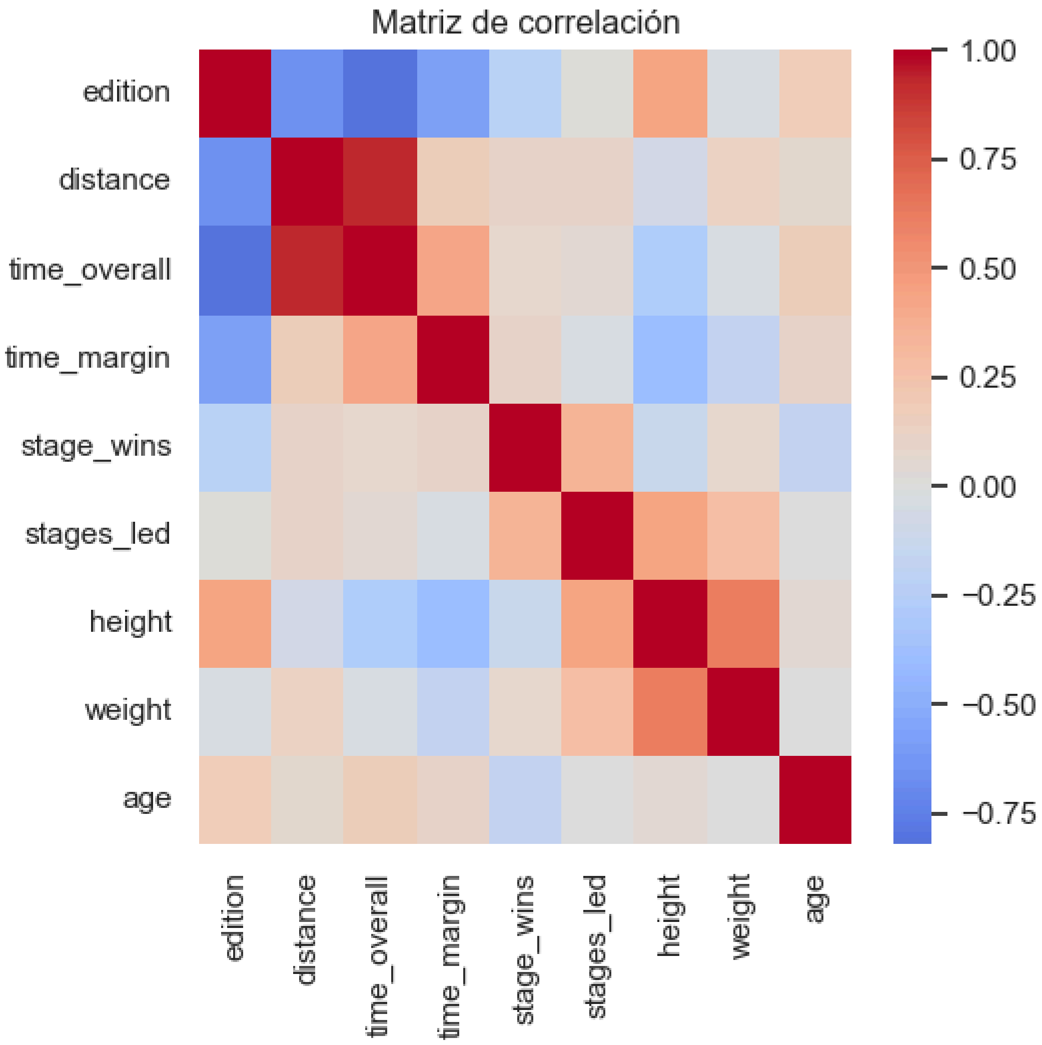

Top 10 Stage
Stage
4
2
11
9
6
10
8
3
7
13
0
50
100
Top 10 Date
Date
1937-07-17
1936-07-31
1939-07-21
1938-07-17
1938-07-30
1937-07-04
1939-07-27
1937-07-22
1938-07-08
1971-06-27
0
1
2
3
Top 10 Origin
Origin
Pau
Bordeaux
Luchon
Paris
Grenoble
Metz
Nice
Perpignan
Briançon
Marseille
0
20
40
60
Top 10 Destination
Destination
Paris
Bordeaux
Pau
Luchon
Metz
Nice
Grenoble
Marseille
Perpignan
Briançon
0
50
100
Top 10 Type
Type
Plain stage
Stage with mountain(s)
Individual time trial
Flat stage
Team time trial
Hilly stage
High mountain stage
Mountain stage
Medium mountain stage
Mountain time trial
0
500
1000
Top 10 Winner
Winner
Eddy Merckx
Mark Cavendish
Bernard Hinault
André Leducq
André Darrigade
Lance Armstrong[n 1]
Nicolas Frantz
François Faber
Jean Alavoine
Charles Pélissier
0
10
20
30


Matriz de correlación
Distance
Distance
1.1
1.0
0.9

```
===== tdf_winners =====
Forma (filas, columnas): (106, 19)

Tipos de datos:
 edition            int64
start_date        object
winner_name       object
winner_team       object
distance         float64
time_overall     float64
time_margin      float64
stage_wins         int64
stages_led         int64
height           float64
weight           float64
age                int64
born              object
died              object
full_name         object
nickname          object
birth_town        object
birth_country     object
nationality       object
dtype: object

Valores faltantes (no-cero):
               na_count  na_pct
full_name           60   56.60
died                50   47.17
height              40   37.74
weight              39   36.79
nickname            32   30.19
time_margin          8    7.55
time_overall         8    7.55

Filas duplicadas: 0

Resumen numérico (describe):
               count         mean          std          min          25%  \
edition       106.0    53.500000   30.743563     1.000000    27.250000
distance      106.0  4212.064151  704.284160  2428.000000  3657.875000
time_overall   98.0   125.754983   41.559391    82.086667    92.601597
time_margin    98.0     0.267727    0.476194     0.002222     0.050833
stage_wins    106.0     2.735849    1.842885     0.000000     1.000000
stages_led    106.0    10.792453    5.307169     1.000000     6.250000
height         66.0     1.778788    0.056989     1.610000     1.740000
weight         67.0    69.253731    6.592795    52.000000    64.500000
age           106.0    27.716981    3.354470    19.000000    26.000000

                      50%          75%          max
edition         53.500000    79.750000   106.000000
distance      4155.500000  4652.500000  5745.000000
time_overall   115.026806   142.678472   238.740278
time_margin      0.101667     0.249931     2.989167
stage_wins       2.000000     4.000000     8.000000
stages_led      12.000000    14.000000    22.000000
```

```
height          1.770000    1.820000    1.900000
weight         69.000000   74.000000   88.000000
age            28.000000   30.000000   36.000000

Sesgo/curtosis:
                  sesgo   curtosis
edition       0.000000  -1.200000
distance      0.161691  -0.313684
time_overall  1.244370   0.713806
time_margin   3.578018  14.667470
stage_wins    0.723573   0.142827
stages_led   -0.011145  -0.822071
height       -0.402901   0.879929
weight        0.029010   0.225412
age          -0.029941  -0.420524

Columnas categóricas y cardinalidad:
  - start_date: 106 únicas
  - winner_name: 63 únicas
  - winner_team: 48 únicas
  - born: 63 únicas
  - died: 38 únicas
  - full_name: 23 únicas
  - nickname: 37 únicas
  - birth_town: 58 únicas
  - birth_country: 15 únicas
  - nationality: 14 únicas
```





Conclusiones

Panorama general

- Los tres datasets se cargaron con detección de codificación robusta y el EDA produjo resúmenes y visualizaciones útiles.
- Se observaron asimetrías y posibles outliers en variables numéricas.
- La correlación entre variables numéricas es moderada; no sugiere multicolinealidad severa en general.
- Donde hubo columna de año, hay tendencias temporales que justifican features por época o interacciones con tipo de etapa.

Conclusiones por dataset

- stage_data

    - Tabla granular por etapa con métricas de desempeño; clave para entender diferencias por tipo de etapa.
    - Útil para construir agregados por año/edición y relacionarlos con ganadores.
- tdf_winners
    - Resumen a nivel ganador/edición.