himalayan_project

October 3, 2025

[1]:
```
import sys
import os
print(sys.executable)
print(os.getcwd())
```

```
C:\Users\mjcd1\anaconda3\envs\unicornenv\python.exe
C:\Users\mjcd1\Desktop\Himalayan_Expeditions\Notebook
```

[2]:
```
import pandas as pd
from IPython.display import display

file_path = "C:\\Users\\mjcd1\\Desktop\\Himalayan_Expeditions\\BBDD\\exped.csv"

df_exped = pd.read_csv(file_path, low_memory=False)
display(df_exped.head(5))
df_exped.info()
```

```
       expid peakid  year  season   host           route1          route2  \
0  ANN260101   ANN2  1960  Spring  Nepal  NW Ridge-W Ridge             NaN
1  ANN269301   ANN2  1969  Autumn  Nepal  NW Ridge-W Ridge             NaN
2  ANN273101   ANN2  1973  Spring  Nepal    W Ridge-N Face             NaN
3  ANN278301   ANN2  1978  Autumn  Nepal    N Face-W Ridge             NaN
4  ANN279301   ANN2  1979  Autumn  Nepal    N Face-W Ridge  NW Ridge of A-IV

  route3 route4      nation  …                          accidents  \
0    NaN    NaN          UK  …                                NaN
1    NaN    NaN  Yugoslavia  …  Draslar frostbitten hands and feet
2    NaN    NaN       Japan  …                                NaN
3    NaN    NaN          UK  …                                NaN
4    NaN    NaN          UK  …                                NaN

  achievment  agency  comrte  stdrte  primrte primmem primref primid   chksum
0        NaN     NaN   False   False    False   False   False    NaN  2442047
1        NaN     NaN   False   False    False   False   False    NaN  2445501
2        NaN     NaN   False   False    False   False   False    NaN  2446797
3        NaN     NaN   False   False    False   False   False    NaN  2448822
4        NaN     NaN   False   False    False   False   False    NaN  2449204

[5 rows x 65 columns]
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11425 entries, 0 to 11424
Data columns (total 65 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   expid       11425 non-null  object
 1   peakid      11425 non-null  object
 2   year        11425 non-null  int64
 3   season      11425 non-null  object
 4   host        11425 non-null  object
 5   route1      11275 non-null  object
 6   route2      360 non-null    object
 7   route3      30 non-null     object
 8   route4      5 non-null      object
 9   nation      11425 non-null  object
 10  leaders     11401 non-null  object
 11  sponsor     10609 non-null  object
 12  success1    11425 non-null  bool
 13  success2    11425 non-null  bool
 14  success3    11425 non-null  bool
 15  success4    11425 non-null  bool
 16  ascent1     2778 non-null   object
 17  ascent2     101 non-null    object
 18  ascent3     11 non-null     object
 19  ascent4     4 non-null      object
 20  claimed     11425 non-null  bool
 21  disputed    11425 non-null  bool
 22  countries   4113 non-null   object
 23  approach    5436 non-null   object
 24  bcdate      9795 non-null   object
 25  smtdate     10670 non-null  object
 26  smttime     4982 non-null   float64
 27  smtdays     9671 non-null   float64
 28  totdays     8406 non-null   float64
 29  termdate    8450 non-null   object
 30  termreason  11425 non-null  object
 31  termnote    4648 non-null   object
 32  highpoint   11425 non-null  int64
 33  traverse    11425 non-null  bool
 34  ski         11425 non-null  bool
 35  parapente   11425 non-null  bool
 36  camps       11425 non-null  int64
 37  rope        11425 non-null  int64
 38  totmembers  11425 non-null  int64
 39  smtmembers  11425 non-null  int64
 40  mdeaths     11425 non-null  int64
 41  tothired    11425 non-null  int64
 42  smthired    11425 non-null  int64
 43  hdeaths     11425 non-null  int64
 44  nohired     11425 non-null  bool
 45  o2used      11425 non-null  bool
 46  o2none      11425 non-null  bool
 47  o2climb     11425 non-null  bool
 48  o2descent   11425 non-null  bool
 49  o2sleep     11425 non-null  bool
 50  o2medical   11425 non-null  bool
 51  o2taken     11425 non-null  bool
 52  o2unkwn     11425 non-null  bool
 53  othersmts   2199 non-null   object
 54  campsites   11046 non-null  object
 55  accidents   3001 non-null   object
 56  achievment  976 non-null    object
 57  agency      9696 non-null   object
 58  comrte      11425 non-null  bool
 59  stdrte      11425 non-null  bool
 60  primrte     11425 non-null  bool
 61  primmem     11425 non-null  bool
 62  primref     11425 non-null  bool
 63  primid      753 non-null    object
 64  chksum      11425 non-null  int64
dtypes: bool(23), float64(3), int64(11), object(28)
memory usage: 3.9+ MB
```

[3]:
```
import pandas as pd
from IPython.display import display

file_path = "C:
 ↪\\Users\\mjcd1\\Desktop\\Himalayan_Expeditions\\BBDD\\himalayan_data_dictionary.
 ↪csv"

df_dictionary = pd.read_csv(file_path, low_memory=False)
display(df_dictionary.head(5))
df_dictionary.info()
```

```
   Table     Field                          Description
0  peaks       NaN                                  NaN
1  peaks    peakid                 Peak ID (primary key)
2  peaks    pkname  Foreign (common) name of the peak
3  peaks   pkname2                Local name of the peak
4  peaks  location        Location of the climbing area

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 165 entries, 0 to 164
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
```

```
 0   Table        165 non-null    object
 1   Field        161 non-null    object
 2   Description  161 non-null    object
dtypes: object(3)
memory usage: 4.0+ KB
```

[4]:
```
import pandas as pd
from IPython.display import display

file_path = "C:\\Users\\mjcd1\\Desktop\\Himalayan_Expeditions\\BBDD\\members.
 ↪csv"

df_members = pd.read_csv(file_path, low_memory=False)
display(df_members.head(5))
df_members.info()
```

```
       expid  membid peakid  myear mseason         fname      lname sex  \
0  AMAD01101       2   AMAD   2001  Spring         Rohan    Buckley   M
1  AMAD01101       1   AMAD   2001  Spring  Marc Cameron   Fairhead   M
2  AMAD01101       3   AMAD   2001  Spring          Mark  Schroeder   M
3  AMAD01101       4   AMAD   2001  Spring         Colin      Smith   M
4  AMAD01101       5   AMAD   2001  Spring         Naomi      Smith   F

      yob    citizen  …  death deathdate deathtime  deathtype  deathhgtm  \
0  1972.0  Australia  …  False       NaN       NaN        NaN          0
1  1968.0  Australia  …  False       NaN       NaN        NaN          0
2  1960.0  Australia  …  False       NaN       NaN        NaN          0
3  1966.0  Australia  …  False       NaN       NaN        NaN          0
4  1970.0  Australia  …  False       NaN       NaN        NaN          0

   deathclass               msmtbid  \
0         NaN          No summit bid
1         NaN  Aborted at high camp
2         NaN  Aborted at high camp
3         NaN          No summit bid
4         NaN          No summit bid

                                          msmtterm  hcn  mchksum
0                   Did not climb or intent to summit  NaN  2439554
1  Bad conditions (deep snow, avalanches, falling…  NaN  2438062
2  Bad conditions (deep snow, avalanches, falling…  NaN  2435183
3                   Did not climb or intent to summit  NaN  2437475
4                   Did not climb or intent to summit  NaN  2438996

[5 rows x 61 columns]

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 89000 entries, 0 to 88999
```

```
Data columns (total 61 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   expid       89000 non-null  object
 1   membid      89000 non-null  int64
 2   peakid      89000 non-null  object
 3   myear       89000 non-null  int64
 4   mseason     89000 non-null  object
 5   fname       88897 non-null  object
 6   lname       87876 non-null  object
 7   sex         89000 non-null  object
 8   yob         83592 non-null  float64
 9   citizen     88993 non-null  object
 10  status      89000 non-null  object
 11  residence   80170 non-null  object
 12  occupation  58261 non-null  object
 13  leader      89000 non-null  bool
 14  deputy      89000 non-null  bool
 15  bconly      89000 non-null  bool
 16  nottobc     89000 non-null  bool
 17  support     89000 non-null  bool
 18  disabled    89000 non-null  bool
 19  hired       89000 non-null  bool
 20  sherpa      89000 non-null  bool
 21  tibetan     89000 non-null  bool
 22  msuccess    89000 non-null  bool
 23  mclaimed    89000 non-null  bool
 24  mdisputed   89000 non-null  bool
 25  msolo       89000 non-null  bool
 26  mtraverse   89000 non-null  bool
 27  mski        89000 non-null  bool
 28  mparapente  89000 non-null  bool
 29  mspeed      89000 non-null  bool
 30  mhighpt     89000 non-null  bool
 31  mperhighpt  64054 non-null  float64
 32  msmtdate1   59731 non-null  object
 33  msmtdate2   449 non-null    object
 34  msmtdate3   22 non-null     object
 35  msmttime1   26512 non-null  float64
 36  msmttime2   244 non-null    float64
 37  msmttime3   7 non-null      float64
 38  mroute1     89000 non-null  int64
 39  mroute2     89000 non-null  int64
 40  mroute3     89000 non-null  int64
 41  mascent1    89000 non-null  int64
 42  mascent2    89000 non-null  int64
 43  mascent3    89000 non-null  int64
 44  mo2used     89000 non-null  bool
 45  mo2none     89000 non-null  bool
 46  mo2climb    89000 non-null  bool
 47  mo2descent  89000 non-null  bool
 48  mo2sleep    89000 non-null  bool
 49  mo2medical  89000 non-null  bool
 50  mo2note     15347 non-null  object
 51  death       89000 non-null  bool
 52  deathdate   1132 non-null   object
 53  deathtime   568 non-null    float64
 54  deathtype   1158 non-null   object
 55  deathhgtm   89000 non-null  int64
 56  deathclass  1158 non-null   object
 57  msmtbid     89000 non-null  object
 58  msmtterm    88824 non-null  object
 59  hcn         295 non-null    float64
 60  mchksum     89000 non-null  int64
dtypes: bool(25), float64(7), int64(10), object(19)
memory usage: 26.6+ MB
```

[5]:
```
import pandas as pd
from IPython.display import display

file_path = "C:\\Users\\mjcd1\\Desktop\\Himalayan_Expeditions\\BBDD\\peaks.csv"

df_peaks = pd.read_csv(file_path, low_memory=False)
display(df_peaks.head(5))
df_peaks.info()
```

```
  peakid         pkname         pkname2  \
0   ACHN         Aichyn  Aychin, Ashvin
1   AMAD     Ama Dablam    Amai Dablang
2   AMOT       Amotsang         Amatson
3   AMPG  Amphu Gyabjen   Amphu Gyabien
4   AMPH        Amphu I             NaN

                                      location  heightm  heightf  \
0              Chandi Himal (SW of Changwathang)     6055    19865
1                                  Khumbu Himal     6814    22356
2                  Damodar Himal (NW of Pokharhan)     6393    20974
3                   Khumbu Himal (N of Ama Dablam)     5630    18471
4  Khumbu Himal (E of Amphu Laptsa, W of Baruntse)     6740    22113

                    himal                      region  open  unlisted  …  \
0  Nalakankar/Chandi/Changla        Kanjiroba-Far West  True     False  …
1                     Khumbu  Khumbu-Rolwaling-Makalu  True     False  …
2                    Damodar    Annapurna-Damodar-Peri  True     False  …
3                     Khumbu  Khumbu-Rolwaling-Makalu  True     False  …
4                     Khumbu  Khumbu-Rolwaling-Makalu  True     False  …
```

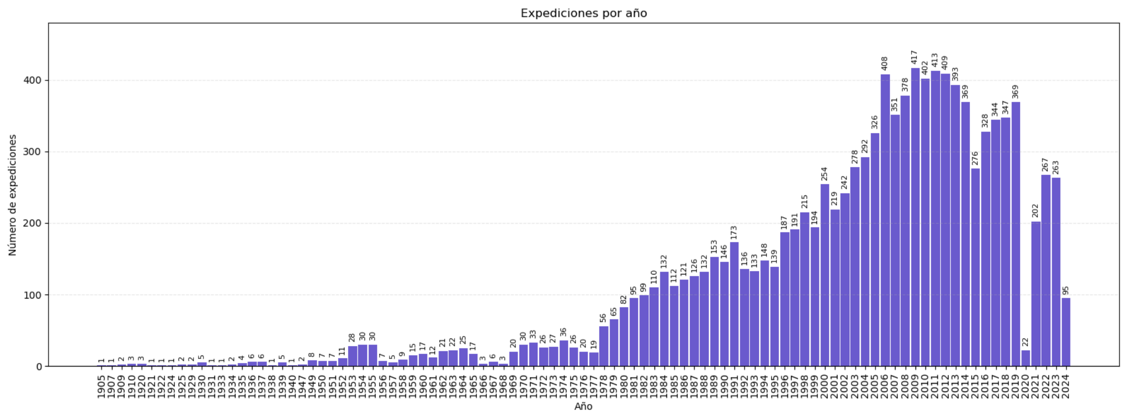
```
        phost  pstatus   pyear pseason pmonth  pday     pexpid  \
0  Nepal only  Climbed  2015.0  Autumn    Sep   3.0  ACHN15301
1  Nepal only  Climbed  1961.0  Spring    Mar  13.0  AMAD61101
2  Nepal only  Climbed  2019.0  Autumn    Oct  24.0  AMOT19301
3  Nepal only  Climbed  1953.0  Spring    Apr  11.0  AMPG53101
4  Nepal only  Climbed  2013.0  Autumn    Oct   9.0  AMPH13301

              pcountry                                         psummiters  \
0                Japan                                 Hiroki Senda, et al
1  New Zealand, USA, UK  Mike Gill, Wally Romanes, Barry Bishop, Michae…
2              Germany                                        Jost Kobusch
3                   UK                              John Hunt, Tom Bourdillon
4              S Korea          An Chi-Young, Kim Young-Mi, Oh Young-Hoon

                  psmtnote
0                      NaN
1                      NaN
2  Possibly climbed earlier
3                      NaN
4                      NaN

[5 rows x 23 columns]
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 23 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   peakid      480 non-null    object
 1   pkname      480 non-null    object
 2   pkname2     257 non-null    object
 3   location    479 non-null    object
 4   heightm     480 non-null    int64
 5   heightf     480 non-null    int64
 6   himal       480 non-null    object
 7   region      480 non-null    object
 8   open        480 non-null    bool
 9   unlisted    480 non-null    bool
 10  trekking    480 non-null    bool
 11  trekyear    29 non-null     float64
 12  restrict    275 non-null    object
 13  phost       480 non-null    object
 14  pstatus     480 non-null    object
 15  pyear       362 non-null    float64
 16  pseason     363 non-null    object
 17  pmonth      357 non-null    object
 18  pday        340 non-null    float64
 19  pexpid      360 non-null    object
 20  pcountry    362 non-null    object
 21  psummiters  477 non-null    object
 22  psmtnote    76 non-null     object
dtypes: bool(3), float64(3), int64(2), object(15)
memory usage: 76.5+ KB
```

[6]:
```python
# Expediciones por año: consulta MySQL, guarda CSV y PNG, y grafica (barras␣
 ↪verticales con más grosor real)

import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sqlalchemy import text
from db_connection import get_engine

# --- 1) Conexión ---
engine = get_engine()

# --- 2) Query ---
sql = text("""
    SELECT `year`, COUNT(*) AS n_expeditions
    FROM himalayan_expeditions.expeditions
    GROUP BY `year`
    ORDER BY `year`;
""")
df = pd.read_sql(sql, engine)

# --- 3) Limpieza y orden ---
df["year"] = pd.to_numeric(df["year"], errors="coerce").astype("Int64")
df = df.dropna(subset=["year"]).astype({"year": "int64"}).sort_values("year")

# --- 4) Guardar CSV ---
os.makedirs("csv", exist_ok=True)
csv_path = os.path.join("csv", "expeditions_por_anio.csv")
df.to_csv(csv_path, index=False)
print(f"CSV guardado en: {csv_path}")

# --- 5) Gráfico (barras verticales más "gruesas" sin pegarse) ---
plt.figure(figsize=(16, 6))
ax = plt.gca()

# separaciones y ancho (ajusta si quieres)
spacing = 1.08       # >1 separa un poco las posiciones en X
width   = 0.95       # ancho de cada barra (no se pegan gracias a spacing)
bar_color = "#6a5acd"  # color (cámbialo si quieres)
```

```
x = np.arange(len(df)) * spacing
bars = ax.bar(x, df["n_expeditions"], width=width, color=bar_color)

ax.set_title("Expediciones por año")
ax.set_xlabel("Año")
ax.set_ylabel("Número de expediciones")

ax.set_xticks(x)
ax.set_xticklabels(df["year"].astype(str), rotation=90, ha="center")

# aire superior y etiquetas verticales con separación del borde
ax.margins(y=0.15)
ax.bar_label(bars, labels=df["n_expeditions"].astype(str),
             padding=4, rotation=90, fontsize=8)

ax.grid(axis="y", linestyle="--", alpha=0.3)
plt.tight_layout()

# --- 6) Guardar imagen ---
os.makedirs("exportados", exist_ok=True)
img_path = os.path.join("exportados", "expeditions_por_anio_vertical.png")
plt.savefig(img_path, dpi=300, bbox_inches="tight")
print(f"PNG guardado en: {img_path}")

plt.show()
```

```
CSV guardado en: csv\expeditions_por_anio.csv
PNG guardado en: exportados\expeditions_por_anio_vertical.png
```

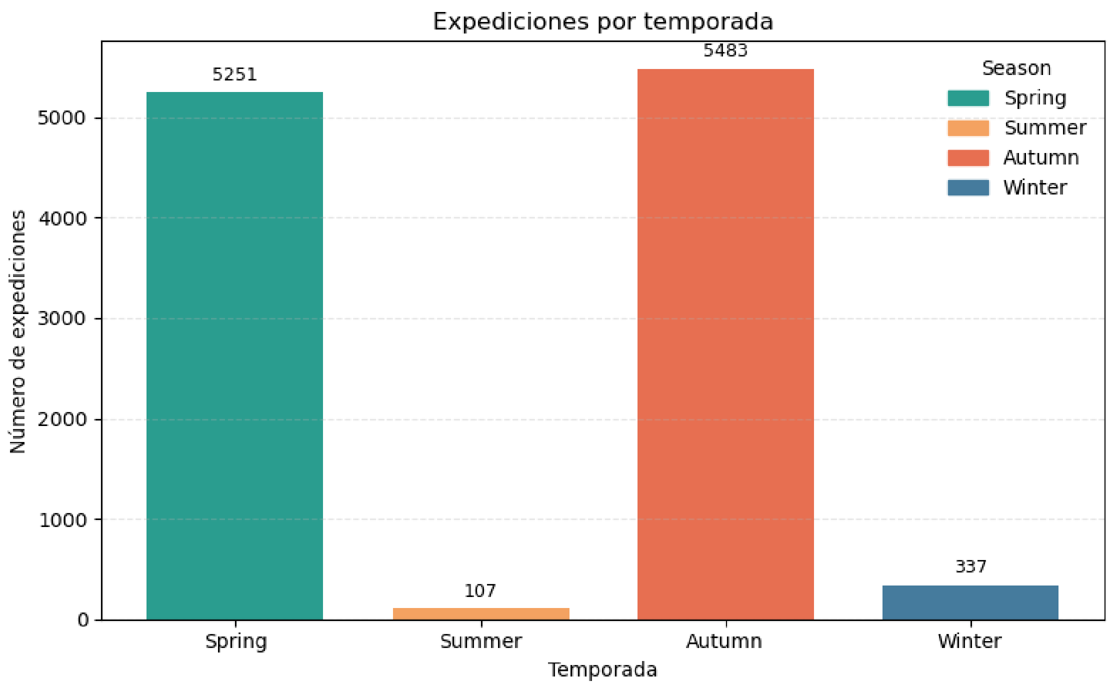


```
[7]: # Expediciones por temporada (season): consulta, CSV, PNG y gráfico con colores␣
     ↪y leyenda
```

```python
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sqlalchemy import text
from db_connection import get_engine
import matplotlib.patches as mpatches

# --- 1) Conexión ---
engine = get_engine()

# --- 2) Query (totales por season) ---
sql = text("""
    SELECT season, COUNT(*) AS n_expeditions
    FROM himalayan_expeditions.expeditions
    WHERE season IS NOT NULL AND season <> ''
    GROUP BY season;
""")
df = pd.read_sql(sql, engine)

# --- 3) Limpieza y orden fijo ---
df["season"] = df["season"].str.strip().str.title()
order = ["Spring", "Summer", "Autumn", "Winter"]
df = df[df["season"].isin(order)].copy()
df["season"] = pd.Categorical(df["season"], categories=order, ordered=True)
df = df.sort_values("season").reset_index(drop=True)

# --- 4) Guardar CSV ---
os.makedirs("csv", exist_ok=True)
csv_path = os.path.join("csv", "expeditions_por_season.csv")
df.to_csv(csv_path, index=False)
print(f"CSV guardado en: {csv_path}")

# --- 5) Gráfico ---
plt.figure(figsize=(8, 5))
ax = plt.gca()

# Colores por temporada (ajústalos si quieres)
palette = {
    "Spring": "#2a9d8f",   # teal
    "Summer": "#f4a261",   # naranja suave
    "Autumn": "#e76f51",   # coral
    "Winter": "#457b9d"    # azul frío
}

x = np.arange(len(df))
```

```python
colors = [palette[s] for s in df["season"]]

bars = ax.bar(x, df["n_expeditions"], color=colors, width=0.72)

ax.set_title("Expediciones por temporada")
ax.set_xlabel("Temporada")
ax.set_ylabel("Número de expediciones")
ax.set_xticks(x)
ax.set_xticklabels(df["season"])

# Etiquetas encima de cada barra
ax.bar_label(bars, labels=df["n_expeditions"].astype(str), padding=4,␣
 ↪fontsize=9)

ax.grid(axis="y", linestyle="--", alpha=0.3)
plt.tight_layout()

# Leyenda con parches de color
handles = [mpatches.Patch(color=palette[s], label=s) for s in order if s in␣
 ↪df["season"].tolist()]
ax.legend(handles=handles, title="Season", frameon=False)

# --- 6) Guardar imagen ---
os.makedirs("exportados", exist_ok=True)
img_path = os.path.join("exportados", "expeditions_por_season.png")
plt.savefig(img_path, dpi=300, bbox_inches="tight")
print(f"PNG guardado en: {img_path}")

plt.show()
```

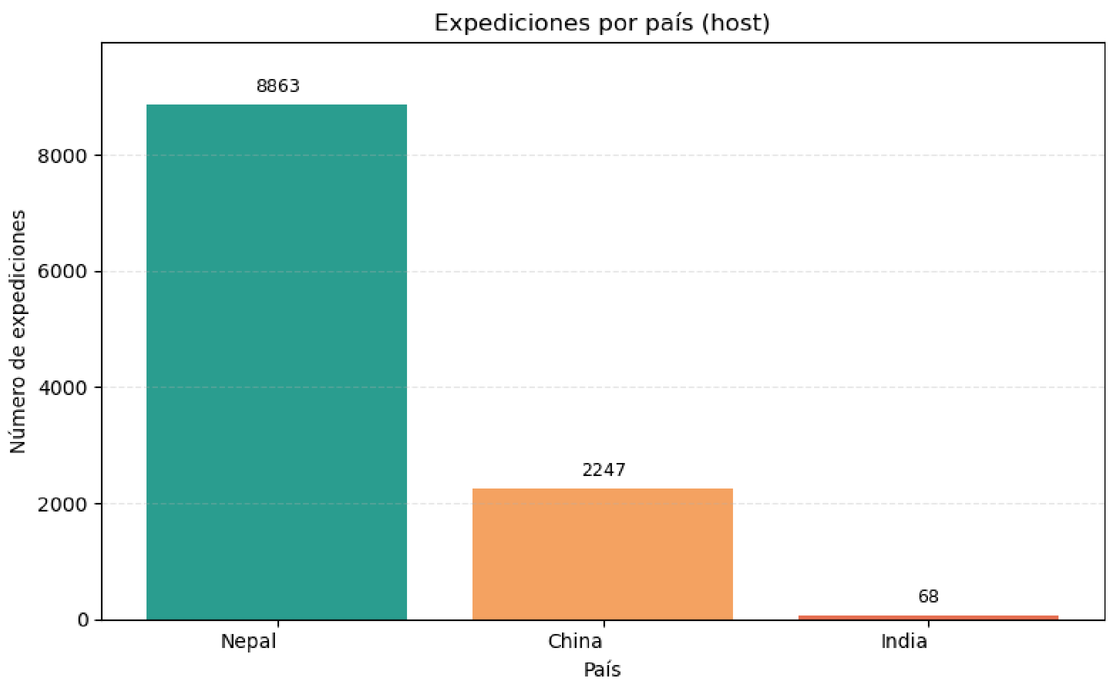
```
CSV guardado en: csv\expeditions_por_season.csv
PNG guardado en: exportados\expeditions_por_season.png
```



[8]:
```
# Expediciones por país (host), excluyendo Unknown/NULL: consulta, CSV, PNG y␣
↪gráfico

import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sqlalchemy import text
from db_connection import get_engine

# 1) Conexión
engine = get_engine()

# 2) Query (host limpio y sin Unknown)
sql = text("""
    SELECT host, n_expeditions FROM (
        SELECT TRIM(host) AS host, COUNT(*) AS n_expeditions
        FROM himalayan_expeditions.expeditions
        WHERE host IS NOT NULL
          AND TRIM(host) <> ''
          AND UPPER(TRIM(host)) <> 'UNKNOWN'
        GROUP BY TRIM(host)
    ) t
    ORDER BY n_expeditions DESC;
```

```
""")
df = pd.read_sql(sql, engine)

# 3) Guardar CSV
os.makedirs("csv", exist_ok=True)
csv_path = os.path.join("csv", "expeditions_por_pais.csv")
df.to_csv(csv_path, index=False)
print(f"CSV guardado en: {csv_path}")

# 4) Gráfico (barras verticales)
plt.figure(figsize=(max(8, len(df)*1.2), 5))
ax = plt.gca()

# Colores por temporada (ajústalos si quieres)
palette = {
    "Nepal": "#2a9d8f",    # teal
    "China": "#f4a261",    # naranja suave
    "India": "#e76f51",    # coral
}

x = np.arange(len(df))
colors = [palette[s] for s in df["host"]]
bars = ax.bar(x, df["n_expeditions"], color=colors, width=0.8)

ax.set_title("Expediciones por país (host)")
ax.set_xlabel("País")
ax.set_ylabel("Número de expediciones")
ax.set_xticks(x)
ax.set_xticklabels(df["host"], rotation=0 if len(df) <= 6 else 45, ha="right")

# Etiquetas numéricas sobre cada barra
ax.margins(y=0.12)
ax.bar_label(bars, labels=df["n_expeditions"].astype(str), padding=4,␣
 ↪fontsize=9)

ax.grid(axis="y", linestyle="--", alpha=0.3)
plt.tight_layout()

# 5) Guardar imagen
os.makedirs("exportados", exist_ok=True)
img_path = os.path.join("exportados", "expeditions_por_pais.png")
plt.savefig(img_path, dpi=300, bbox_inches="tight")
print(f"PNG guardado en: {img_path}")

plt.show()
```

```
CSV guardado en: csv\expeditions_por_pais.csv
PNG guardado en: exportados\expeditions_por_pais.png
```



```
[9]: # ================================
# Mapa interactivo por pico/métrica (Opción A - barra compacta)
# ================================

import os, re
import pandas as pd
import plotly.express as px
import plotly.io as pio
import plotly.graph_objects as go
import ipywidgets as widgets
from ipywidgets import Layout as L # Importar L para layouts más claros
from IPython.display import display, HTML, Image
from sqlalchemy import text
import pycountry

from db_connection import get_engine

pio.renderers.default = "notebook"  # estable en VS Code/Anaconda

engine = get_engine()
os.makedirs("csv", exist_ok=True)
os.makedirs("exportados", exist_ok=True)

# *************************************************************
# AÑADE ESTO: CSS para eliminar márgenes del entorno del notebook
# ***************************************************************
display(HTML("""<style>
.widget-subarea, .output_subarea {
    padding: 0 !important;
    margin: 0 !important;
    max-width: none !important;
}
.jupyter-widgets.widget-box {
    margin: 0 !important;
    padding: 0 !important;
}
</style>"""))


# -----------------------------------
# Picos y escalas de color sugeridas
# -----------------------------------
PEAKS = [
    ("Everest",       "LOWER(p.pkname) = 'everest'",             "Plasma"),
    ("Kangchenjunga", "p.peakid IN ('KANG','KANC','KANN','KANS')",     ␣
 ↪"Magma"),
    ("Lhotse",        "LOWER(p.pkname) LIKE 'lhotse%'",          "Viridis"),
    ("Makalu",        "LOWER(p.pkname) LIKE 'makalu%'",          "Cividis"),
    ("Manaslu",       "LOWER(p.pkname) LIKE 'manaslu%'",          "Inferno"),
]

# -----------------------------
# Normalización a ISO-3
# -----------------------------
SPECIALS = {
    "UK":"GBR","U.K.":"GBR","USA":"USA","U.S.A.":"USA","S Korea":"KOR","N␣
 ↪Korea":"PRK",
    "W Germany":"DEU","Czech Republic":"CZE","Russia":"RUS","Nepal":
 ↪"NPL","China":"CHN",
    "India":"IND","Japan":"JPN","New Zealand":"NZL","Australia":"AUS","Spain":
 ↪"ESP",
    "France":"FRA","Italy":"ITA","Switzerland":"CHE","Poland":"POL","Austria":
 ↪"AUT",
    "Germany":"DEU","Slovenia":"SVN"
}
def to_iso3(name:str):
    if not isinstance(name,str): return None
    name = name.strip()
    if not name or name.upper()=="UNKNOWN": return None
    if name in SPECIALS: return SPECIALS[name]
    try: return pycountry.countries.lookup(name).alpha_3
    except: return None

# -----------------------------
# SQL base (parametrizada)
# -----------------------------
SQL_BASE = """
SELECT
    TRIM(e.nation)                        AS nation,
    {metrica_sql}                         AS value,
    MIN(e.year)                           AS year_min,
    MAX(e.year)                           AS year_max
FROM himalayan_expeditions.expeditions e
JOIN himalayan_expeditions.peaks p ON p.peakid = e.peakid
WHERE p.pkname <> '[placeholder]'
    AND e.nation IS NOT NULL
    AND TRIM(e.nation) <> ''
    AND UPPER(TRIM(e.nation)) <> 'UNKNOWN'
    AND ({cond})
GROUP BY TRIM(e.nation)
HAVING value > 0
ORDER BY value DESC;
"""

def get_df(peak_cond:str, metric:str) -> pd.DataFrame:
    """metric in {'expeditions','deaths'}"""
    metrica_sql = "COUNT(*)" if metric=="expeditions" else "COALESCE(SUM(e.
  ↪mdeaths),0)"
    sql = text(SQL_BASE.format(metrica_sql=metrica_sql, cond=peak_cond))
    df = pd.read_sql(sql, engine)
    if df.empty: return df
    df["iso3"] = df["nation"].apply(to_iso3)
    df = df.dropna(subset=["iso3"]).reset_index(drop=True)
    total = df["value"].sum()
    df["share"] = df["value"]/total
    return df

# -----------------------------
# Widgets
# -----------------------------
peak_dropdown = widgets.Dropdown(
    options=[lbl for (lbl,_,_) in PEAKS], value="Everest", description="Pico:"
)
metric_toggle = widgets.ToggleButtons(
    options=[("Expediciones","expeditions"), ("Muertes","deaths")],
    value="expeditions", description="Métrica:"
)
topn_slider = widgets.IntSlider(
    value=15, min=5, max=30, step=1, description="Top-N:",␣
 ↪continuous_update=False
)
btn_csv = widgets.Button(description="CSV", icon="save", button_style="info")
btn_png = widgets.Button(description="PNG", icon="image",␣
 ↪button_style="warning")

# ************************************************************
# AJUSTE DE LAYOUTS PARA COMPACTAR
# ************************************************************
# Usamos L (Layout) para asegurar que se usa el objeto correcto
peak_dropdown.layout = L(width="200px", margin="0 10px 0 0")
metric_toggle.style.button_width = "150px"
metric_toggle.layout = L(width="220px", margin="0 10px 0 0")
# CLAVE: Margen a la derecha del slider para separarlo de los botones
topn_slider.layout = L(width="350px", margin="0 40px 0 0")
btn_csv.layout = L(width="80px", margin="0 5px 0 0")
btn_png.layout = L(width="80px", margin="0")

# ************************************************************
# BARRA DE CONTROL CORREGIDA: HBox ÚNICO (sin spacer)
# ************************************************************
controls = widgets.HBox(
    [peak_dropdown, metric_toggle, topn_slider, btn_csv, btn_png],
    layout=L(
        width="100%",
        # CLAVE: flex-start pega todos los widgets a la izquierda
        justify_content="flex-start",
        align_items="center",
        margin="0", padding="0"
    )
)

out = widgets.Output(layout=L(margin="0", padding="0")) # Asegurar que el␣
 ↪output no tiene margen
display(controls, out)

# estado para exportación
_last_df = {"df": None, "label": "", "metric": "", "fig": None}

# -----------------------------
# Render y exportadores
# -----------------------------
def actualizar(_=None):
    with out:
        out.clear_output()
        label, cond, colorscale = next(t for t in PEAKS if t[0]==peak_dropdown.
↪value)
        metric = metric_toggle.value

        df = get_df(cond, metric)
        title = f" {'Expediciones' if metric=='expeditions' else 'Muertes'}␣
↪por país - {label}"

        if df.empty:
            print(f"{title}\n\n(No hay datos para mostrar.)")
            _last_df.update({"df": None, "label": label, "metric": metric,␣
↪"fig": None})
            return

        fig = px.choropleth(
            df, locations="iso3", locationmode="ISO-3", color="value",
            hover_name="nation", color_continuous_scale=colorscale, title=title
        )
        fig.update_geos(showcountries=True, showcoastlines=True,␣
↪projection_type="natural earth")

        # CLAVE: Asegurar que el margen izquierdo del gráfico es 0 (o muy bajo)
        fig.update_layout(margin={"r":0,"t":60,"l":5,"b":0}, width=1000,␣
↪height=600)

        # tooltips enriquecidos (sin f-strings con %{...})
        unidad = "expediciones" if metric=="expeditions" else "muertes"
        fig.update_traces(
            hovertemplate=(
                "<b>%{hovertext}</b><br>"
                + unidad + ": %{z:.0f}<br>"
                + "participación: %{customdata[0]:.1%}<br>"
                + "años activos: %{customdata[1]}-%{customdata[2]}<extra></
↪extra>"
            ),
            customdata=df[["share","year_min","year_max"]].to_numpy()
        )

        # Top-N con borde y ranking lateral
        topn = max(5, min(int(topn_slider.value), len(df)))
        df_top = df.nlargest(topn, "value").copy()

        fig.add_trace(go.Choropleth(
            locations=df_top["iso3"],
            z=df_top["value"],
            locationmode="ISO-3",
            colorscale=colorscale,
            showscale=False,
            marker_line_color="black",
            marker_line_width=1.2,
            hovertext=df_top["nation"],
            hovertemplate=(
                "<b>%{hovertext}</b><br>"
                + unidad + ": %{z:.0f}<extra>Top-" + str(topn) + "</extra>"
            ),
            name=f"Top-{topn}"
        ))

        rank_text = "<br>".join(
            f"{i+1}. {r.nation} - {int(r.value)}"
            for i, r in df_top.reset_index(drop=True).iterrows()
        )
        fig.add_annotation(
            x=1.02, y=0.5, xref="paper", yref="paper", showarrow=False,
            align="left", bgcolor="rgba(255,255,255,0.75)", bordercolor="#ccc",
            text=f"<b>Top-{topn}</b><br>{rank_text}"
        )

        fig.show()
        _last_df.update({"df": df.copy(), "label": label, "metric": metric,␣
 ↪"fig": fig})

def export_csv(_):
    st = _last_df
    if st["df"] is None: return
    slug_label = re.sub(r"[^a-z0-9]+","_", st["label"].lower())
    slug_metric = "exped" if st["metric"]=="expeditions" else "deaths"
    path = os.path.join("csv", f"mapa_{slug_label}_{slug_metric}.csv")
    st["df"][["nation","iso3","value","share","year_min","year_max"]].
 ↪to_csv(path, index=False)
    print(f"CSV exportado: {path}")

def export_png(_):
    st = _last_df
    if st["fig"] is None: return
    slug_label = re.sub(r"[^a-z0-9]+","_", st["label"].lower())
    slug_metric = "exped" if st["metric"]=="expeditions" else "deaths"
    path = os.path.join("exportados", f"mapa_{slug_label}_{slug_metric}.png")
    st["fig"].write_image(path, width=1000, height=600, scale=1)  # requiere␣
 ↪kaleido
    display(Image(path))

# enlazar eventos
```

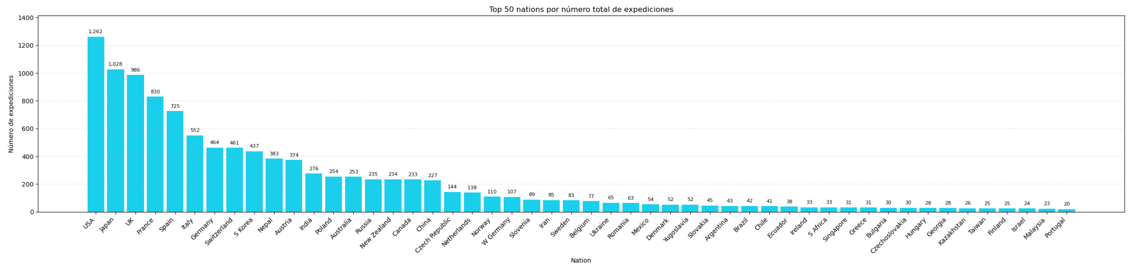
```
peak_dropdown.observe(actualizar, names="value")
metric_toggle.observe(actualizar, names="value")
topn_slider.observe(actualizar, names="value")
btn_csv.on_click(export_csv)
btn_png.on_click(export_png)

# primera renderización
actualizar()
```

```
<IPython.core.display.HTML object>

HBox(children=(Dropdown(description='Pico:', layout=Layout(margin='0 10px 0 0',␣
↪width='200px'), options=('Ever…

Output(layout=Layout(margin='0', padding='0'))
```

[10]:
```
# Top-50 nations por número de expediciones (totales) - barras verticales color␣
↪#1ACFEB

import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sqlalchemy import text
from db_connection import get_engine

# 1) Conexión
engine = get_engine()

# 2) Query
sql = text("""
    SELECT TRIM(nation) AS nation, COUNT(*) AS n_expeditions
    FROM himalayan_expeditions.expeditions
    WHERE nation IS NOT NULL
      AND TRIM(nation) <> ''
      AND UPPER(TRIM(nation)) <> 'UNKNOWN'
    GROUP BY TRIM(nation)
    ORDER BY n_expeditions DESC
    LIMIT 50;
""")
df = pd.read_sql(sql, engine)

# 3) Guardar CSV
os.makedirs("csv", exist_ok=True)
csv_path = os.path.join("csv", "top50_nations_por_expediciones.csv")
df.to_csv(csv_path, index=False)
print(f"CSV guardado en: {csv_path}")
```

```
# 4) Gráfico
plt.figure(figsize=(max(12, len(df)*0.5), 6))
ax = plt.gca()

bar_color = "#1ACFEB"
x = np.arange(len(df))
bars = ax.bar(x, df["n_expeditions"], color=bar_color, width=0.85)

ax.set_title("Top 50 nations por número total de expediciones")
ax.set_xlabel("Nation")
ax.set_ylabel("Número de expediciones")

ax.set_xticks(x)
ax.set_xticklabels(df["nation"], rotation=45, ha="right")

# Etiquetas con miles separados
ax.margins(y=0.12)
ax.bar_label(bars, labels=[f"{int(v):,}" for v in df["n_expeditions"]],␣
 ↪padding=4, fontsize=8)

ax.grid(axis="y", linestyle="--", alpha=0.3)
plt.tight_layout()

# 5) Guardar PNG
os.makedirs("exportados", exist_ok=True)
img_path = os.path.join("exportados", "top50_nations_por_expediciones.png")
plt.savefig(img_path, dpi=300, bbox_inches="tight")
print(f"PNG guardado en: {img_path}")

plt.show()
```

```
CSV guardado en: csv\top50_nations_por_expediciones.csv
PNG guardado en: exportados\top50_nations_por_expediciones.png
```



[14]:
```
# ==============================
# Éxitos por año (H vs M) - líneas
# ==============================
```

```python
import os
import pandas as pd
import plotly.express as px
import ipywidgets as w
from ipywidgets import Layout as L
from sqlalchemy import text
from IPython.display import display, HTML
from db_connection import get_engine

engine = get_engine()

# Pico -> peakid(s) (incluye variantes)
PEAKS = {
    "Todos": [],
    "Everest": ["EVER"],
    "Kangchenjunga": ["KANG", "KANC", "KANN", "KANS"],
    "Lhotse": ["LHOT"],
    "Makalu": ["MAKA"],
    "Manaslu": ["MANA", "MANN"],  # <- ojo: MANA y MANN
}

def _sql_in_list(codes):
    if not codes:
        return ""  # sin filtro de pico
    quoted = ",".join([f"'{c}'" for c in codes])
    return f" AND m.peakid IN ({quoted}) "

def query_success_by_year(peak_codes=None, year_min=1900, year_max=2025):
    peak_filter = _sql_in_list(peak_codes or [])
    sql = f"""
        SELECT
            m.myear AS year,
            CASE WHEN m.sex='M' THEN 'Hombres'
                 WHEN m.sex='F' THEN 'Mujeres'
                 ELSE 'Desconocido' END AS sexo,
            COUNT(*) AS intentos,
            SUM(
                CASE
                  WHEN m.msmtdate1 IS NOT NULL AND TRIM(m.msmtdate1) <> ''
                  THEN 1 ELSE 0
                END
            ) AS exitos
        FROM himalayan_expeditions.members m
        WHERE m.myear BETWEEN :ymin AND :ymax
          AND m.sex IN ('M','F')
          {peak_filter}
        GROUP BY m.myear, sexo
        HAVING year IS NOT NULL
        ORDER BY year;
    """
    df = pd.read_sql(text(sql), engine, params={"ymin": int(year_min), "ymax":␣
↪int(year_max)})
    if df.empty:
        return df
    df["pct_exito"] = (df["exitos"] / df["intentos"]).replace([np.inf, np.nan],␣
↪0) * 100
    return df



# ---------- Controles ----------
peak_dd = w.Dropdown(options=list(PEAKS.keys()), value="Everest",␣
↪description="Pico:",
                     layout=L(width="220px"))
metric_tb = w.ToggleButtons(options=[("Éxitos","exitos"), ("% éxito","pct")],
                            value="exitos", description="Métrica:",
                            layout=L(width="220px"))
year_range = w.IntRangeSlider(value=[1950, 2025], min=1900, max=2025, step=1,
                              description="Años:", readout=True,
                              layout=L(width="420px"))
btn_csv = w.Button(description="CSV", icon="file", tooltip="Exportar CSV",
                   layout=L(width="70px"), button_style="info")
btn_png = w.Button(description="PNG", icon="image", tooltip="Exportar PNG",
                   layout=L(width="70px"), button_style="warning")

topbar = w.HBox([peak_dd, metric_tb, year_range, btn_csv, btn_png],
                layout=L(width="100%", justify_content="space-between"))

out_fig = w.Output()
out_tbl = w.Output()
display(topbar, out_fig, out_tbl)

_last_df = None
_last_fig = None
_last_peak = None
_last_metric = None

def render(*_):
    global _last_df, _last_fig, _last_peak, _last_metric

    peak_name = peak_dd.value
    codes = PEAKS[peak_name]
    met = metric_tb.value
    y0, y1 = year_range.value
    df = query_success_by_year(codes, y0, y1)

    with out_fig:
        out_fig.clear_output(wait=True)

        if df.empty:
            fig = px.line(title=f"No hay datos para {peak_name} en {y0}-{y1}")
            fig.show()
            _last_df = None
            _last_fig = fig
            _last_peak = peak_name
            _last_metric = met
            return

        # Pivot para líneas por sexo
        if met == "exitos":
            ycol = "exitos"; ytitle = "Éxitos (cumbres)"
        else:
            ycol = "pct_exito"; ytitle = "% éxito"
        pivot = df.pivot(index="year", columns="sexo", values=ycol).fillna(0)

        fig = px.line(pivot.reset_index(), x="year", y=pivot.columns,
                      title=f"Ascensos exitosos por año - {peak_name}",
                      labels={"value": ytitle, "year":"Año", "variable":"Sexo"},
                      markers=True)
        fig.update_layout(legend_title="Sexo", height=520, margin=dict(l=20,␣
↪r=30, t=60, b=40))
        if met == "pct_exito":
            fig.update_yaxes(ticksuffix=" %")

        fig.show()

    with out_tbl:
        out_tbl.clear_output(wait=True)
        show_cols = ["year","sexo","intentos","exitos","pct_exito"]
        df_show = df[show_cols].copy()
        df_show.rename(columns={"year":"Año","sexo":"Sexo","intentos":
↪"Intentos",
                                "exitos":"Éxitos","pct_exito":"% Éxito"},␣
↪inplace=True)
        display(df_show.style.format({"% Éxito":"{:.2f}"}))

    # para exportar
    _last_df = df_show
    _last_fig = fig
    _last_peak = peak_name
    _last_metric = met

def on_csv(_):
    if _last_df is None:
        return
    os.makedirs("csv", exist_ok=True)
    fname = f"csv/exitos_{_last_metric}_{_last_peak}_{year_range.
 ↪value[0]}-{year_range.value[1]}.csv".replace(" ","_")
    _last_df.to_csv(fname, index=False)
    print(f"CSV guardado en: {fname}")

def on_png(_):
    if _last_fig is None:
        return
    os.makedirs("exportados", exist_ok=True)
    fname = f"exportados/exitos_{_last_metric}_{_last_peak}_{year_range.
 ↪value[0]}-{year_range.value[1]}.png".replace(" ","_")
    try:
        _last_fig.write_image(fname, width=1100, height=550, scale=2)
        print(f"PNG guardado en: {fname}")
    except Exception as e:
        print("Para exportar PNG instala kaleido: pip install -U kaleido")
        print(e)

btn_csv.on_click(on_csv)
btn_png.on_click(on_png)

peak_dd.observe(render, names="value")
metric_tb.observe(render, names="value")
year_range.observe(render, names="value")

render()
```

```
HBox(children=(Dropdown(description='Pico:', index=1,␣
 ↪layout=Layout(width='220px'), options=('Todos', 'Everest…
```

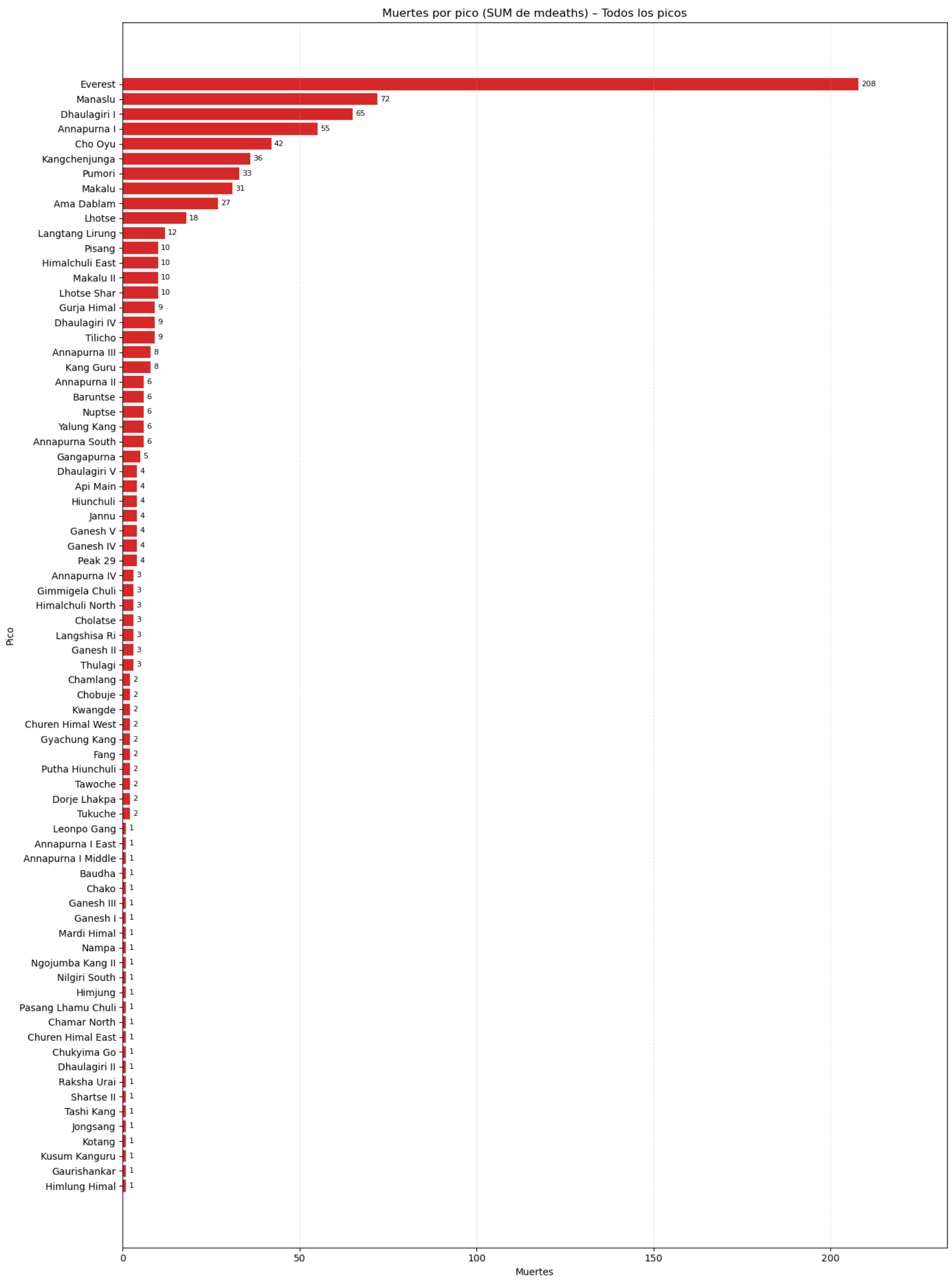
```
Output()
```

```
Output()
```

[12]:
```
# Muertes en TODOS los picos (SUM de mdeaths) + CSV + gráfico horizontal
# Opción A: etiquetar TODAS las barras con bar_label

import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sqlalchemy import text
```

```python
from db_connection import get_engine

engine = get_engine()
os.makedirs("csv", exist_ok=True)
os.makedirs("exportados", exist_ok=True)

# 1) Traer todas las muertes por pico
sql = text("""
    SELECT
        p.peakid,
        p.pkname,
        COALESCE(SUM(e.mdeaths),0) AS deaths
    FROM himalayan_expeditions.expeditions e
    JOIN himalayan_expeditions.peaks p ON p.peakid = e.peakid
    WHERE p.pkname <> '[placeholder]'
    GROUP BY p.peakid, p.pkname
    HAVING deaths > 0
    ORDER BY deaths DESC
""")
df = pd.read_sql(sql, engine)

# 2) Guardar CSV
csv_path = os.path.join("csv", "muertes_por_pico__mdeaths_ALL.csv")
df.to_csv(csv_path, index=False)
print(f"CSV guardado en: {csv_path}  (filas: {len(df):,})")

# 3) Gráfico horizontal con etiquetas en TODAS las barras
if df.empty:
    print("No hay datos de muertes para graficar.")
else:
    n = len(df)
    plt.figure(figsize=(14, max(6, 0.25*n)))  # alto proporcional al # de picos
    ax = plt.gca()
    y = np.arange(n)

    bars = ax.barh(y, df["deaths"], color="#d62828")  # rojo
    ax.set_yticks(y)
    ax.set_yticklabels(df["pkname"])
    ax.invert_yaxis()  # mayor arriba

    ax.set_title("Muertes por pico (SUM de mdeaths) - Todos los picos")
    ax.set_xlabel("Muertes")
    ax.set_ylabel("Pico")
    ax.grid(axis="x", linestyle="--", alpha=0.3)

    # margen a la derecha y etiquetas para TODAS las barras
    xmax = df["deaths"].max()
    ax.set_xlim(0, xmax * 1.12)
    ax.bar_label(bars,
                 labels=df["deaths"].astype(int).astype(str),
                 padding=3, fontsize=8)

    plt.tight_layout()

    # 4) Guardar PNG
    img_path = os.path.join("exportados", "muertes_por_pico__mdeaths_ALL.png")
    plt.savefig(img_path, dpi=300, bbox_inches="tight")
    print(f"PNG guardado en: {img_path}")

    plt.show()
```

```
CSV guardado en: csv\muertes_por_pico__mdeaths_ALL.csv  (filas: 75)
PNG guardado en: exportados\muertes_por_pico__mdeaths_ALL.png
```



[13]:
```
# =========================
# Dashboard picos por nación
```

```python
# ==========================
import os
import pandas as pd
import numpy as np
import plotly.express as px
from sqlalchemy import text
import ipywidgets as w
from ipywidgets import Layout as L
from IPython.display import display, HTML
from db_connection import get_engine

# ****************
#  NUEVA ADICIÓN
# ****************
display(HTML("""<style>
.widget-subarea, .output_subarea {
    padding: 0 !important;
    margin: 0 !important;
    max-width: none !important;
}
.jupyter-widgets.widget-box {
    margin: 0 !important;
    padding: 0 !important;
}
</style>"""))


engine = get_engine()

PEAKS = {
    "Everest": ["EVER"],
    "Kangchenjunga": ["KANG", "KANC", "KANN", "KANS"],
    "Lhotse": ["LHOT"],
    "Makalu": ["MAKA"],
    "Manaslu": ["MANA", "MANN"],
}

def _sql_in_str(codes):
    return ",".join([f"'{c}'" for c in codes])

#  Ahora solo usamos expeditions
def query_expeditions_fatality(peak_codes, min_attempts=10, topn=15):
    codes = _sql_in_str(peak_codes)
    sql = f"""
    WITH stats AS (
        SELECT
            TRIM(e.nation) AS nation,
            SUM(e.totmembers + e.tothired) AS attempts,
            SUM(e.mdeaths + e.hdeaths) AS deaths
        FROM himalayan_expeditions.expeditions e
        WHERE e.peakid IN ({codes})
        GROUP BY TRIM(e.nation)
    )
    SELECT
        nation,
        attempts,
        deaths,
        ROUND((deaths / NULLIF(attempts,0)) * 100, 2) AS fatality_pct
    FROM stats
    WHERE nation IS NOT NULL AND TRIM(nation) <> ''
      AND attempts >= :min_attempts
    ORDER BY fatality_pct DESC, deaths DESC
    LIMIT :topn;
    """
    return pd.read_sql(text(sql), engine, params={"min_attempts":␣
  ↪int(min_attempts), "topn": int(topn)})

# ====== Definición de controles ======
peak_dd = w.Dropdown(options=list(PEAKS.keys()), value="Everest",␣
  ↪description="Pico:")
topn_slider = w.IntSlider(value=15, min=5, max=30, step=1, description="Top-%:")
min_attempts_slider = w.IntSlider(value=10, min=1, max=100, step=1,␣
  ↪description="Mín. intentos:")

btn_csv = w.Button(description="CSV", icon="save", button_style="info")
btn_png = w.Button(description="PNG", icon="image", button_style="warning")

peak_dd.layout = L(width="220px", margin="0 20px 0 0")
topn_slider.layout = L(width="240px", margin="0 20px 0 0")
min_attempts_slider.layout = L(width="240px", margin="0 20px 0 0")
btn_csv.layout = L(width="80px", margin="0 10px 0 20px")   # ← más ancho +␣
  ↪separación
btn_png.layout = L(width="80px", margin="0 10px 0 0")

# ====== Barra superior corregida ======
topbar = w.HBox(
    [peak_dd, topn_slider, min_attempts_slider, btn_csv, btn_png],
    layout=L(width="100%", align_items="center", margin="0 0 15px 0",␣
  ↪padding="15px 0")
)

# ====== Outputs ======
out_fig   = w.Output(layout=L(margin="0", padding="0"))
out_table = w.Output(layout=L(margin="0", padding="0"))
```

```
dashboard_container = w.VBox([topbar, out_fig, out_table],
                             layout=L(width='100%', margin='0', padding='0',␣
 ↪border='none'))
display(dashboard_container)

# ---- Render ----
_last_df = None
_last_fig = None
_last_peak = None

def make_fatality_figure(df, peak_name, topn):
    if df.empty:
        fig = px.bar(title=f"No hay datos para {peak_name}")
        return fig, df

    df_plot = df.head(int(topn)).copy()
    df_plot = df_plot.sort_values("fatality_pct", ascending=True)

    fig = px.bar(
        df_plot,
        x="nation", y="fatality_pct",
        color="fatality_pct",
        color_continuous_scale="RdPu",
        text="fatality_pct",
        title=f"Miembros - % de fatalidad por nación (Top-{len(df_plot)}) -␣
 ↪{peak_name}",
    )
    fig.update_traces(texttemplate="%{text:.1f}%")
    fig.update_layout(
        xaxis_title="Nación", yaxis_title="% fatalidad",
        coloraxis_colorbar_title="% fatalidad",
        margin=dict(l=15, r=40, t=60, b=60),
        height=520,
    )
    return fig, df_plot

def render(*_):
    global _last_df, _last_fig, _last_peak

    peak_name = peak_dd.value
    codes = PEAKS[peak_name]
    topn = int(topn_slider.value)
    min_att = int(min_attempts_slider.value)

    df = query_expeditions_fatality(codes, min_attempts=min_att, topn=topn)
    fig, df_plot = make_fatality_figure(df, peak_name, topn)
    df_tab = df[["nation", "attempts", "deaths", "fatality_pct"]].copy()
    df_tab.rename(columns={
        "nation": "Nación",
        "attempts": "Intentos",
        "deaths": "Muertes",
        "fatality_pct": "Fatalidad_%"
    }, inplace=True)

    _last_df = df_tab.copy()
    _last_fig = fig
    _last_peak = peak_name

    with out_fig:
        out_fig.clear_output(wait=True)
        fig.show()

    with out_table:
        out_table.clear_output(wait=True)
        sty = df_tab.style.format({"Fatalidad_%": "{:.2f}"})
        try:
            sty = sty.hide(axis="index")
        except Exception:
            pass
        display(sty)

def on_export_csv(_):
    if _last_df is None:
        return
    os.makedirs("csv", exist_ok=True)
    fname = f"csv/fatalidad_{_last_peak}_por_nacion.csv".replace(" ", "_")
    _last_df.to_csv(fname, index=False)
    print(f"CSV guardado en: {fname}")

def on_export_png(_):
    if _last_fig is None:
        return
    os.makedirs("exportados", exist_ok=True)
    fname = f"exportados/fatalidad_{_last_peak}_por_nacion.png".replace(" ",␣
  ↪"_")
    try:
        _last_fig.write_image(fname, width=1100, height=550, scale=2)
        print(f"PNG guardado en: {fname}")
    except Exception as e:
        print("Para exportar PNG instala kaleido:  pip install -U kaleido")
        print(f"Detalle: {e}")
```

```
btn_csv.on_click(on_export_csv)
btn_png.on_click(on_export_png)

# ---- Eventos + primer render ----
peak_dd.observe(render, names="value")
topn_slider.observe(render, names="value")
min_attempts_slider.observe(render, names="value")

render()
```

```
<IPython.core.display.HTML object>
```

```
VBox(children=(HBox(children=(Dropdown(description='Pico:',␣
 ↪layout=Layout(margin='0 20px 0 0', width='220px'),…
```

```
[ ]:
```