Data Analytics

Himalayan Expeditions Analysis

Camille NIVAULT-RUIZ for

April, 2024

Table of content

1- Introduction

Since the first ascent of Everest by Edmund Hillary and Tenzing Norgay in 1953, the appeal of mountaineering and alpinism has continuously grown. The Himalaya, in particular, has become a focal point for adventurers from around the world. Technological advances in climbing equipment have made these once inaccessible peaks more attainable, leading to now infamous scenes of overcrowding on the slopes of Everest:

This increasing influx raises crucial safety questions and poses major logistical challenges. The numbers speak for themselves: while there were only about ten expeditions per year on Everest in the 1970s, today there are more than 200 annually, representing over 3000 climbers in 2019. This exponential increase creates complex problems in managing camps, supplies, and most importantly, environmental impact, especially in terms of waste management :

- 2016 -

UNION OF HIMALAYAN AGENCIES

Faced with these challenges, the main trekking agencies in the region have taken the initiative to unite by creating the **Union of Himalayan Agencies**. Their mission is to anticipate and provide concrete solutions to safety and overcrowding issues. With this in mind, the Union has commissioned me for a freelance mission with a clear objective: **Analyze the success and risk factors of Himalayan expeditions to improve the planning and safety of future expeditions.**

This study aims to provide crucial data for more sustainable and secure expedition management, thus preserving the beauty and integrity of these mythical peaks for future generations.

Deliverables:

1. Interactive data exploration dashboard for the UHA (Union of Himalayan Agencies) This dashboard will allow the UHA to freely and deeply explore the available data.
2. Risk and success rate prediction application This application, based on a machine learning model

These deliverables will provide the UHA with powerful tools to:

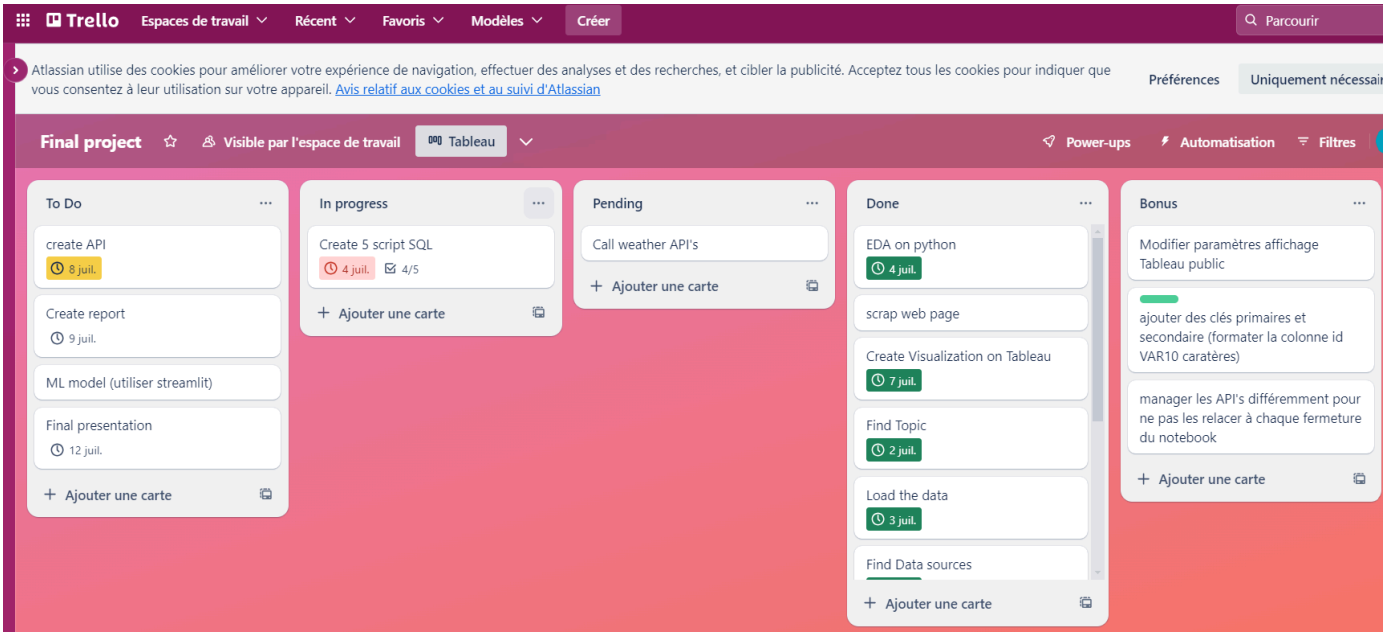
- Make informed decisions based on detailed historical data
- Plan safer expeditions with a higher likelihood of success
- Identify emerging trends and areas requiring special attention
- Develop long-term strategies for sustainable management of Himalayan expeditions

2- Project Management

Creating a Trello board to manage the project's progress day by day with 5 lists :

- To Do : remains to be done
- In progress : current tasks
- Pending : blocked tasks (difficulties)
- Done : completed tasks
- Bonus : optional finishes to do if I have time

Using the Kanban method to manage tasks daily. Using deadlines to manage priorities and maintain control over planning.


Trello
Espaces de travail
Récent
Favoris
Modèles
Créer
Parcourir
Atlassian utilise des cookies pour améliorer votre expérience de navigation, effectuer des analyses et des recherches, et cibler la publicité. Acceptez tous les cookies pour indiquer que vous consentez à leur utilisation sur votre appareil. Avis relatif aux cookies et au suivi d'Atlassian
Préférences
Uniquement nécessai
Final project
Visible par l'espace de travail
Tableau
Power-ups
Automatisation
Filtres
To Do
create API
8 juil.
Create report
9 juil.
ML model (utiliser streamlit)
Final presentation
12 juil.
Ajouter une carte
In progress
Create 5 script SQL
4 juil.
4/5
Ajouter une carte
Pending
Call weather API's
Ajouter une carte
Done
EDA on python
4 juil.
scrap web page
Create Visualization on Tableau
7 juil.
Find Topic
2 juil.
Load the data
3 juil.
Find Data sources
Ajouter une carte
Bonus
Modifier paramètres affichage Tableau public
ajouter des clés primaires et secondaire (formater la colonne id VAR10 caratères)
manager les API's différemment pour ne pas les relacer à chaque fermeture du notebook
Ajouter une carte

3- Data and data sources

For this final project, I collected data from various sources to create my final dataset.
I used methods like web scraping to extract data from websites, gathered a large dataset in Kaggle.com and connected to a platform through API to obtained additional Data.
I thus obtained a complete dataset which allowed me to successfully carry out this project.

★ Flat file :

I found a dataset available on Kaggle about Himalaya Expeditions.
The Himalayan Database is a compilation of records for all expeditions that have climbed in the Nepal Himalaya. The database is based on the expedition archives of Elizabeth Hawley, a longtime journalist based in Kathmandu, and it is supplemented by information gathered from books, alpine journals and correspondence with Himalayan climbers.

The data cover all expeditions from 1905 through Spring 2019 to more than 465 significant peaks in Nepal. Also included are expeditions to both sides of border peaks such as Everest, Cho Oyu, Makalu and Kangchenjunga as well as to some smaller border peaks. Data on expeditions to trekking peaks are included for early attempts, first ascents and major accidents.

The database contains 3 tables constructed as below :

peaks		
Columns	Description	Data type
peak_id	Unique identifier for each peak	text
peak_name	Name of peak	text
peak_alternative_name	Alternative name of peak	text
height_metres	altitude	bigint
climbing_status	Climbed / unclimbed	text
first_ascent_year	Year of first ascension	double
first_ascent_country	Country of first ascension	text

first_ascent_expedition_id	Expedition_id of first ascension	text

Expedition		
Columns	Description	Data type
expedition_id	Unique identifier for each expedition	text
peak_id	Unique identifier for each peak	text
peak_name	Name of the peak	text
year	year	bigint
season	season	text
basecamp_date	Date from basecamp	datetime
highpoint_date	Date where the team was in the highpoint	datetime
termination_date	End date of the expedition	date
termination_reason	Reason of termination	text
highpoint_metres	Max altitude	double
members	Number of climber	bigint
member_deaths	Number of climber death	bigint
hired_staff	Number of staff	bigint
hired_staff_deaths	Number of staff death	bigint
oxygen_used	use of oxygen	tinyint(1)
trekking_agency	Name of the trekking agency	text
is_success	Success / fail	tinyint(1)
harmonized_agency_name	Harmonized trekking agency name	varchar(255)

members		
Columns	Description	Data type
expedition_id	Unique identifier for each expedition	text
member_id	Unique identifier for each member	text
peak_id	Unique identifier for each peak	text
peak_name	Name of the peak	text

year	year	bigint
season	season	text
sex	Sex of the climber	text
age	Age of the climber	double
citizenship	nationality	text
expedition_role	Role in the expedition	text
hired	Member of staff	tinyint(1)
highpoint_metres	Max altitude	double
success	success	tinyint(1)
solo	solo	tinyint(1)
oxygen_used	use of oxygen	tinyint(1)
died	Died	tinyint(1)
death_cause	Cause of death	text
death_height_metres	Death altitude	double
injured	injured	tinyint(1)
injury_type	Type of injury	text
injury_height_metres	Injury altitude	double

★ API :

Nominatim API was used as an additional resource for my analysis to retrieve latitude and longitude from peak names. By collecting GPS coordinates of the peaks, visualizations on maps can be generated.

The collected data was directly added to the 'Peaks' dataframe.

The initial idea was, once the coordinates were obtained, to use the latitude and longitude to fetch weather data via another API. Unfortunately, most APIs found are either paid for historical data or have very limited daily query allowances.

Python code :

```
# API Nominatim Option 1

geolocator = Nominatim(user_agent="my_himalayan_expedition_app", timeout=10)

# Fonction pour obtenir les coordonnées
```

```
def get_coordinates(address):
    try:
        location = geolocator.geocode(address)
        if location:
            return location.latitude, location.longitude
        else:
            return None, None
    except GeocoderTimedOut:
        print("Timeout occurred. Retrying...")
        time.sleep(1)
        return get_coordinates(address)  # Retry once after a short delay
    except GeocoderServiceError as e:
        print(f"Geocoding service error: {e}")
        return None, None
# Ajoutez les colonnes pour les coordonnées
peaks_df['latitude'] = None
peaks_df['longitude'] = None
# Obtenez les coordonnées pour chaque sommet et mettez à jour le DataFrame
for index, row in peaks_df.iterrows():
    peak = row['peak_name']
    lat, lon = get_coordinates(peak)
    if lat is not None and lon is not None:
        peaks_df.at[index, 'latitude'] = lat
        peaks_df.at[index, 'longitude'] = lon
        print(f"{peak}: {lat}, {lon}")
    else:
        print(f"{peak}: coordinates not found")
```

★ Web scraping

The Topchinatravel website was used to scrape temperature and weather tables by season for Mount Everest. The collected data was directly saved into CSV files.

Two tables were created:

season_climate		
Columns	Description	Data type
Type	month	string
Celsius	degree in celsius	float
Fahrenheit	degree in fahrenheit	float
peak_id	Unique identifier for each peak	string

weather_period_everest		
Columns	Description	Data type
seaon	season	string
climate	Most frequent climate	string
windows_date	Date windows from season	string
peak_id	Unique identifier for each peak	string

4- Data cleaning

Here are the steps taken for the 3 dataframes to clean the data before starting the exploratory data analysis:

1- shape :

```
print (f'expedition : {expedition_df.shape}')
print (f'members : {members_df.shape}')
print (f'peaks : {peaks_df.shape}')
```

```
expedition : (10364, 16)
members : (76519, 21)
peaks : (468, 8)
```

2- check null values :

```
print (f'expedition : {expedition_df.isnull().sum()}')
print (f'members : {members_df.isnull().sum()}')
print (f'peaks : {peaks_df.isnull().sum()}')
```

```
expedition : expedition_id            0
peak_id                   0
peak_name                 1
year                      0
season                    0
basecamp_date          1095
highpoint_date          650
termination_date       2380
termination_reason        0
highpoint_metres        414
```

3- check duplicate value :

```
print (f'expedition : {expedition_df.duplicated().sum()}')
print (f'peaks : {peaks_df.duplicated().sum()}')
print (f'members : {members_df.duplicated().sum()}')
```

```
expedition : 0
peaks : 0
members : 0
```

4- Formatting datetime :

```
expedition_df['basecamp_date'] = pd.to_datetime(expedition_df['basecamp_date'],
errors='coerce')
expedition_df['highpoint_date'] = pd.to_datetime(expedition_df['highpoint_date'],
errors='coerce')
expedition_df['termination_date'] = pd.to_datetime(expedition_df['termination_date'],
errors='coerce')
expedition_df['is_success'] = expedition_df['termination_reason'].apply(lambda x: True
if x == 'Success (main peak)' else False)
print(expedition_df.dtypes)
expedition_df.head()
```

5- check data type :

```
print(peaks_df.dtypes)
```

```
peak_id                     object
peak_name                   object
peak_alternative_name       object
height_metres                int64
climbing_status             object
first_ascent_year          float64
first_ascent_country        object
first_ascent_expedition_id  object
location                    object
region                      object
dtype: object
```

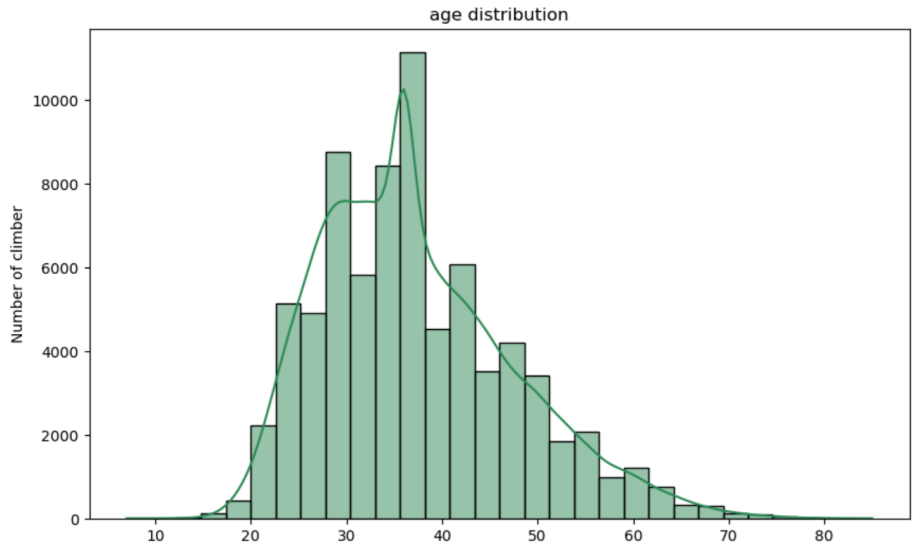
6- Managing null values : this is an example of managing null value in expedition_df

```
# Replace null value by median for numerical column
numeric_cols = expedition_df.select_dtypes(include=['float64', 'int64']).columns
expedition_df[numeric_cols] =
expedition_df[numeric_cols].fillna(expedition_df[numeric_cols].median())

# Replace null value by "Unknown" for categorical column
categorical_cols = expedition_df.select_dtypes(include=['object']).columns
expedition_df[categorical_cols] = expedition_df[categorical_cols].fillna("Unknown")

# manage null basecamp_date (avg date per year + season)
mean_basecamp_date = expedition_df.groupby(['year',
'season'])['basecamp_date'].transform('mean')
mean_basecamp_date = mean_basecamp_date.dt.floor('D')
expedition_df['basecamp_date'] = pd.to_datetime(expedition_df['basecamp_date'],
errors='coerce')

# manage null hightpoint_date (avf diff)
mean_highpoint_diff = (expedition_df['highpoint_date'] -
expedition_df['basecamp_date']).dt.days.mean()
```

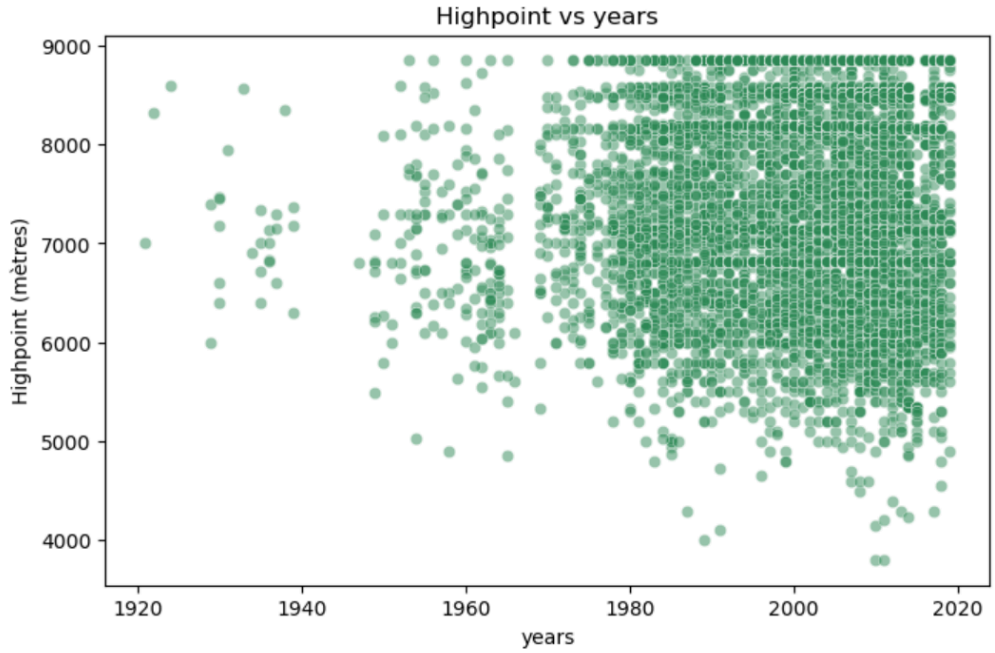
```
expedition_df['highpoint_date'].fillna(expedition_df['basecamp_date'] +
pd.to_timedelta(mean_highpoint_diff, unit='D'), inplace=True)

# manage null termination_date (avg diff)
mean_termination_diff = (expedition_df['termination_date'] -
expedition_df['basecamp_date']).dt.days.mean()
expedition_df['termination_date'].fillna(expedition_df['basecamp_date'] +
pd.to_timedelta(mean_termination_diff, unit='D'), inplace=True)
expedition_df['termination_date'] = pd.to_datetime(expedition_df['termination_date'],
errors='coerce')
expedition_df['termination_date'] = expedition_df['termination_date'].dt.date

# drop null value
expedition_df = expedition_df.dropna()
print (f'expedition : {expedition_df.isnull().sum()}')
expedition_df
```

5- Exploratory data analysis

★ EDA on python :

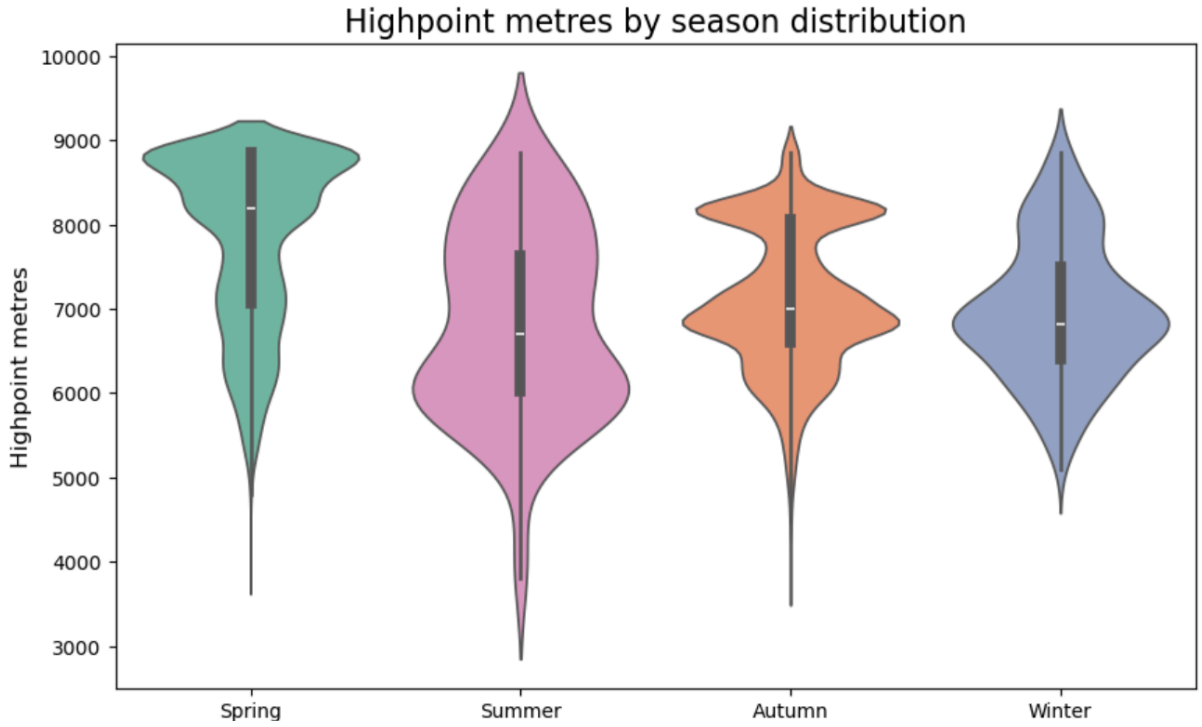


⇒ The age distribution of climbers reveals a right-skewed distribution, with the majority of climbers concentrated between the ages of 30 and 40.



⇒ The scatter plot depicts the relationship between the highest points reached during expeditions and the years in which they occurred. There is a noticeable increase in the number

of expeditions reaching higher altitudes post-1970, reflecting advancements in climbing technology and increased interest in high-altitude mountaineering.



⇒ The violin plot illustrates the distribution of highpoint meters achieved across different seasons. Notably, the highest points are typically reached during the summer and spring seasons, with winter expeditions generally achieving lower highpoints. This pattern suggests that summer and spring provide more favorable conditions for reaching higher altitudes.

★ Statistics null hypothesis :

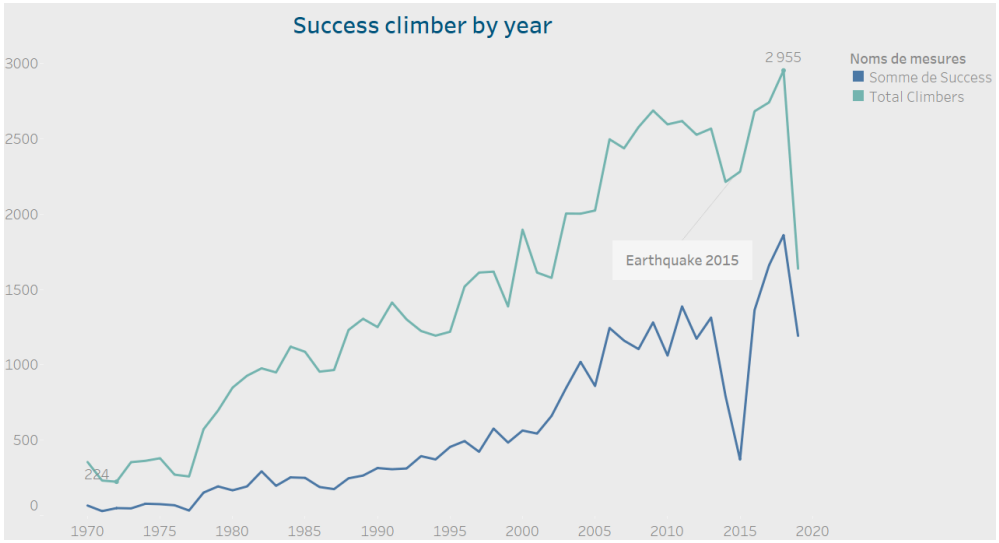
Verification of the hypothesis that men and women reach the same average altitude (using a 2-sample T-test) :

```
# Two Sample T-test
df_female = members_df[members_df["sex"]=="F"]["highpoint_metres"]
df_male = members_df[members_df["sex"]=="M"]["highpoint_metres"
#Set the hypothesis
#H0: highpoint male = highpoint female
#H1: highpoint male != highpoint female

#significance level = 0.05
result = st.ttest_ind(df_male,df_female, equal_var=False)
p_value = result.pvalue
if p_value > 0.05:
    print("We are not able to reject the null hypothesis")
```

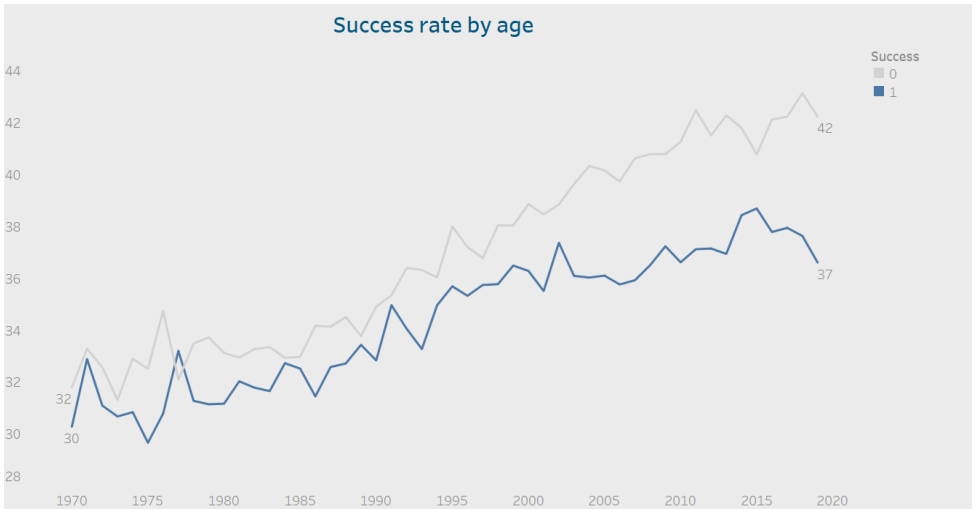
```
else:
    print("We reject the null hypothesis")
```

⇒ means the data provides strong evidence that there is a difference in the average altitudes reached by men and women.

★ EDA on Tableau :

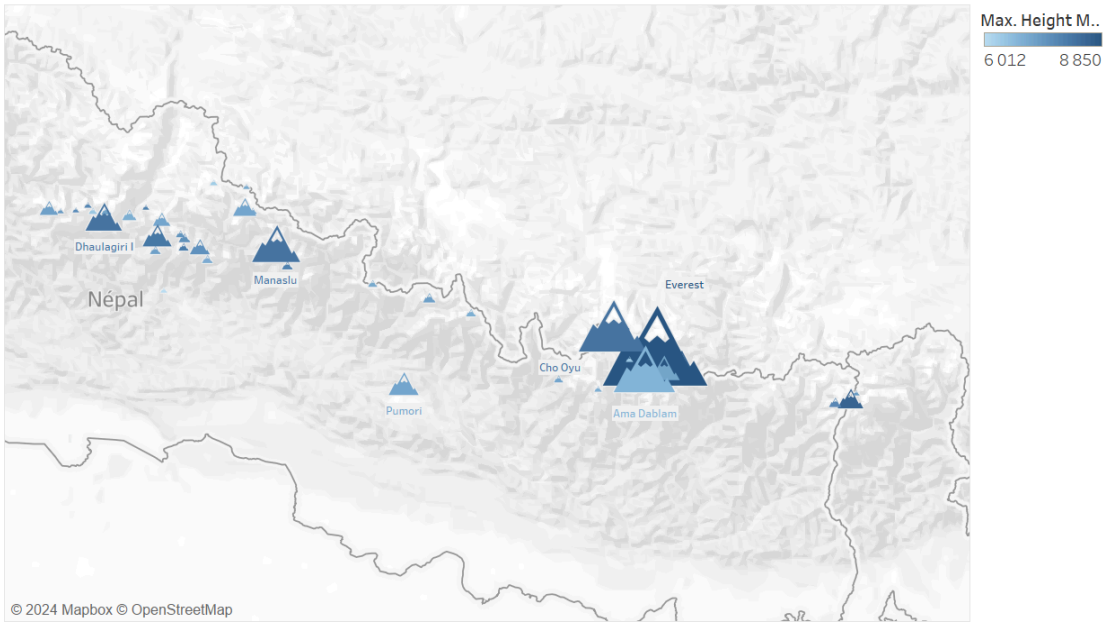


⇒ The number of climbers as well as successful climbs has seen a significant increase since 1970, reaching a peak in 2019, highlighting the growing popularity of climbing among the general public.

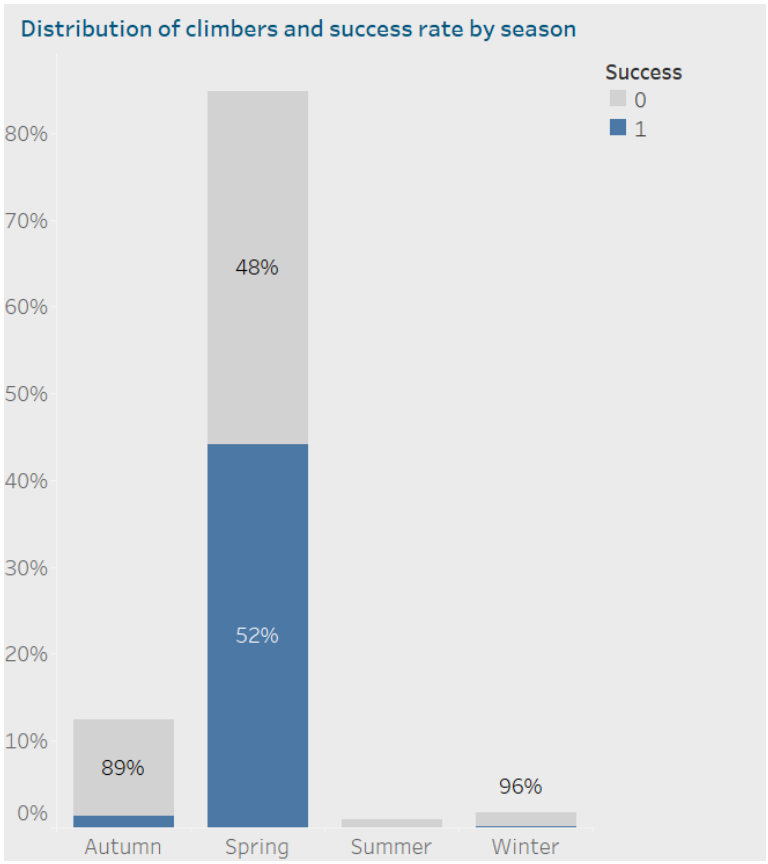


⇒ Here we can observe a clear difference in the average age between expeditions that succeeded (37 years) and those that failed (42 years). This significant gap suggests that good physical condition is necessary for high-altitude mountaineering and indicates a correlation between the age of the climber and the outcome of the expedition.

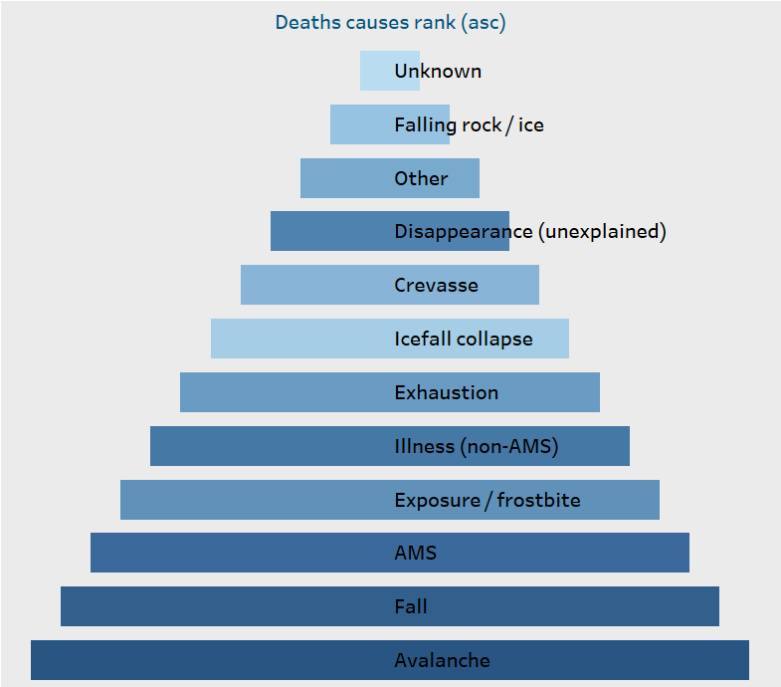
Maps



⇒ This map shows the top 50 most climbed mountains in the Himalayas. A darker color indicates higher peaks (based on altitude), and the size of the icon indicates how frequently the mountains are climbed.



⇒ This stacked bar chart shows the distribution of seasons for expeditions and the success rate by season for Everest. Here, we can see that the most favorable season is summer, with over 85% of expeditions and a success rate of 52%.



⇒ Here we can see the ranking of causes of death across the 50 most climbed mountains in the Himalayas between 1970 and 2019. Avalanches, falls, and AMS (Acute Mountain Sickness) are the leading causes of death.

A Tableau Public story has been published containing all visualizations. It consists of 2 dashboards:

- One provides general information on all the data (with optional filters by year and/or mountains).
- The other dashboard offers details on a specific peak: risk factors and success factors during ascents. This helps improve planning and safety for future expeditions.

Link :
https://public.tableau.com/views/Himalaya-expeditions-Story/Histoire1?:language=fr-FR&:sid=&:redirect=auth&:display_count=n&:origin=viz_share_link

6- Database type selection

When it comes to choice of database type, since I have structured data organized into interrelated tables with a predefined schema, relationships, and foreign keys it seemed appropriate to use a Relational Database. This type of database will help me to reduce data redundancy and improve data integrity, easily manipulate data, and of course, use SQL to perform queries that involve joining data from multiple Tables.

Relational Databases using SQL are a type of database management system that organizes data into structured tables with predefined relationships between them.

Key Characteristics:

- Tables: Data is stored in tables (relations) with rows and columns.
- Relationships: Tables can be linked through keys (primary and foreign keys).
- Structured data: Data follows a predefined schema.

Advantages:

- Data integrity and consistency
- Complex queries and joins
- Standardization (SQL)
- Scalability

7- Database creation

1- I created database on mySQL by the code :
CREATE DATABASE himalaya

2- The tables were created in Python with an engine connection as follows:

```python
pw = os.getenv('my_SQL_pw')
pw = urllib.parse.quote_plus(pw)
```

✓ 0.0s Python

```python
connection_string = 'mysql+pymysql://root:' + pw + '@localhost:3306/'
engine = create_engine(connection_string)
```

✓ 0.0s Python

```python
expedition_df.to_sql(name='expedition', con=engine, schema='himalaya', index=False, if_exists='replace')
members_df.to_sql(name='members', con=engine, schema='himalaya', index=False, if_exists='replace')
peaks_df.to_sql(name='peaks', con=engine, schema='himalaya', index=False, if_exists='replace')
weather_period_everest.to_sql(name='weather_period', con=engine, schema='himalaya', index=False, if_exists='replace')
season_climate.to_sql(name='season_climate', con=engine, schema='himalaya', index=False, if_exists='replace')
```

✓ 5.6s Python

3- The database is ready to be queried in MySQL.

4- If I find extra time, I would like to correct the schema of my tables as follows:

- Add primary and foreign keys during the database creation
- Format the data type of the columns

8- SQL Queries

Some examples of queries that were executed (all queries available in repository) :

1 - Select some peak details to expose it in the API (route : /peaks/<peak_id>) :

```
SELECT
p.peak_name,
p.peak_id,
p.height_metres,
p.peak_alternative_name,
p.first_ascent_year,
p.first_ascent_country,
sum(m.died) as nb_died,
sum(m.success) as nb_success,
sum(m.success)/count(m.success) *100 as tx_success,
sum(m.oxygen_used)/count(m.oxygen_used) *100 as tx_oxygen_used,
avg(age) as age_mean
FROM peaks p
LEFT JOIN members m on p.peak_id = m.peak_id
WHERE p.peak_id = "HIUP"
GROUP BY p.peak_name, p.peak_id, p.height_metres, p.peak_alternative_name, p.first_ascent_year, p.first_ascent_country
ORDER BY peak_id
```

2- Create table with all of death detail :

```
Create table death
SELECT
expedition_id,
member_id,
m.peak_id,
m.peak_name,
year,
season,
sex,
age,
citizenship,
hired,
highpoint_metres,
solo,
oxygen_used,
death_cause,
death_height_metres,
p.longitude,
p.latitude
FROM himalaya.members m
LEFT JOIN himalaya.peaks p ON m.peak_id = p.peak_id
WHERE died = 1
```

expedition_id	member_id	peak_id	peak_name	year	season	sex	age	citizenship	hired	highpoint_metres	solo	oxygen_used	death_cause
AMAD79302	AMAD79302-04	AMAD	Ama Dablam	1979	Autumn	M	23	New Zealand	0	6100	0	0	Avalanche
AMAD83301	AMAD83301-01	AMAD	Ama Dablam	1983	Autumn	M	31	Switzerland	0	7400	0	0	Fall
AMAD83301	AMAD83301-13	AMAD	Ama Dablam	1983	Autumn	F	28	Switzerland	0	7400	0	0	Fall
AMAD85102	AMAD85102-03	AMAD	Ama Dablam	1985	Spring	M	32	Japan	0	6814	0	0	Fall
AMAD88102	AMAD88102-04	AMAD	Ama Dablam	1988	Spring	M	33	Canada	0	6300	0	0	Fall
AMAD92102	AMAD92102-01	AMAD	Ama Dablam	1992	Spring	M	36	Spain	0	6814	0	0	Fall
ANN170101	ANN170101-05	ANN1	Annapurna I	1970	Spring	M	32	UK	0	7315	0	0	Falling rock / ice

3- Selected most popular trekking agency to expose it in the API (route : /statistics):

```sql
SELECT
harmonized_agency_name,
count(distinct expedition_id) as nb_expedition,
sum(is_success)/count(distinct expedition_id)  as success_rate
FROM himalaya.expedition
WHERE harmonized_agency_name != 'Unknown'
GROUP BY harmonized_agency_name
ORDER BY nb_expedition desc
```

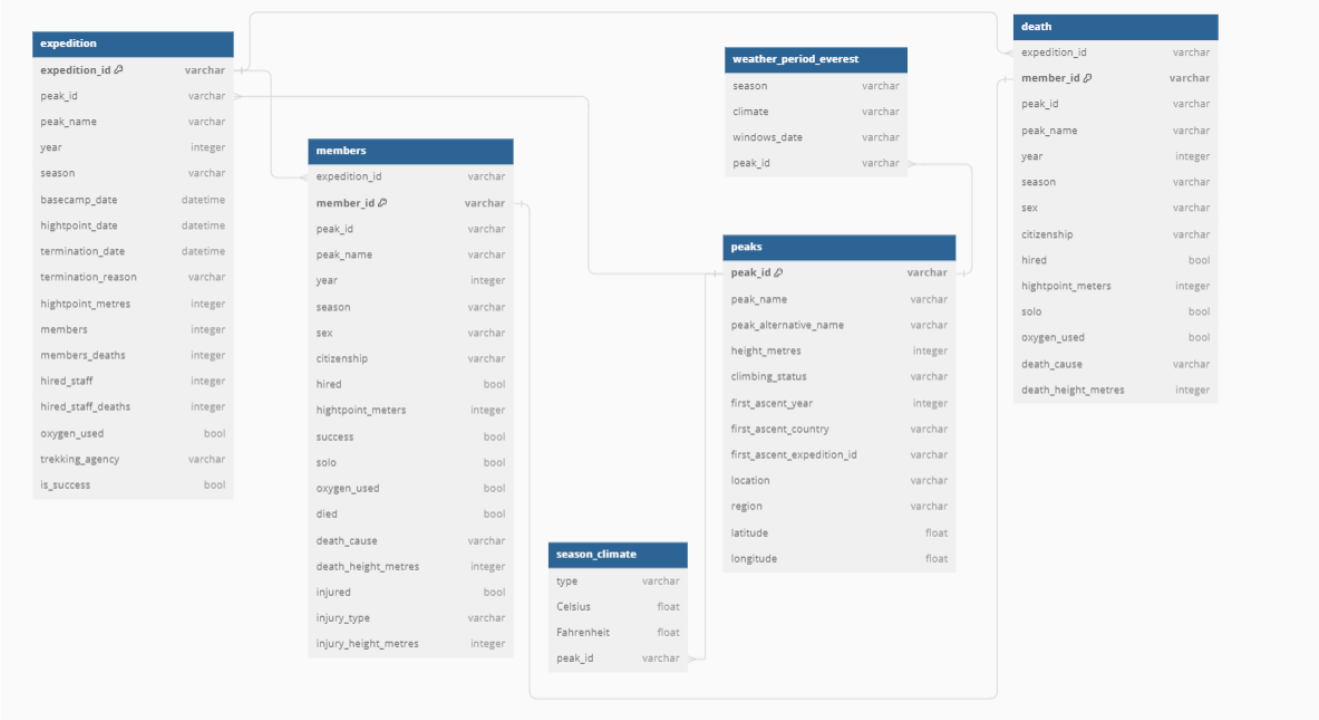
4- Harmonized trekking agency_name (from table expedition) :

```sql
ALTER TABLE himalaya.expedition ADD COLUMN harmonized_agency_name VARCHAR(255);

SET SQL_SAFE_UPDATES = 0;
-- Mettre à jour les valeurs de la nouvelle colonne
UPDATE himalaya.expedition
SET harmonized_agency_name = TRIM(
    REGEXP_REPLACE(
        REGEXP_REPLACE(
            trekking_agency,
            '\\([^)]*\\)',  -- Supprimer tout ce qui est entre parenthèses
            ''
        ),
        '[/?].*$',  -- Supprimer tout ce qui suit un / ou un ?
        ''
    )
);
SET SQL_SAFE_UPDATES = 1;
SELECT distinct trekking_agency, harmonized_agency_name FROM himalaya.expedition ;
```

trekking_agency	harmonized_agency_name
Adventure 6000	Adventure 6000
Adventure Alternative Nepal	Adventure Alternative Nepal
Adventure Ascent (Happy Feet permit)	Adventure Ascent
Adventure Ascents	Adventure Ascents
Adventure Extreme	Adventure Extreme
Adventure Extreme (Himalayan Ascent permit)	Adventure Extreme
Adventure Extreme (Monterosa permit)	Adventure Extreme

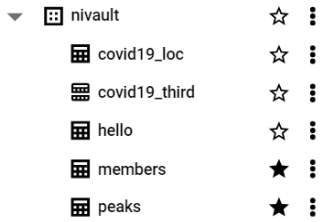
9- Entity Relationship Diagram (ERD)

Tables have primary keys and are connected through foreign keys either by a relationship “one to many”, or by a relationship one to one.



10- Big Query

I created 2 tables “members” and “peaks” on Big Query via python code.



Preview (peaks):

Row	peak_id	peak_name	peak_alternative_name	height_metres	climbing_status	first_ascent_yea	first_ascent_country
1	CHEO	Cheo Himal	Unknown	6820	Climbed	1991.0	Japan, Nepal
2	GYAJ	Gyajikang	Gyaji Khang	7074	Climbed	1994.0	Japan, Nepal
3	HIML	Himlung Himal	Unknown	7126	Climbed	1992.0	Japan, Nepal
4	KGUR	Kang Guru	Naurgaon	6981	Climbed	1955.0	W Germany
5	NEMJ	Nemjung	Unknown	7140	Climbed	1983.0	Japan, Nepal
6	RATC	Ratna Chuli	Unknown	7035	Climbed	1996.0	Japan, Nepal
7	HIMJ	Himjung	Nemjung Goth	7092	Climbed	2012.0	S Korea

Test with simple query, number of climbers per year and percentage change compared to the previous year :

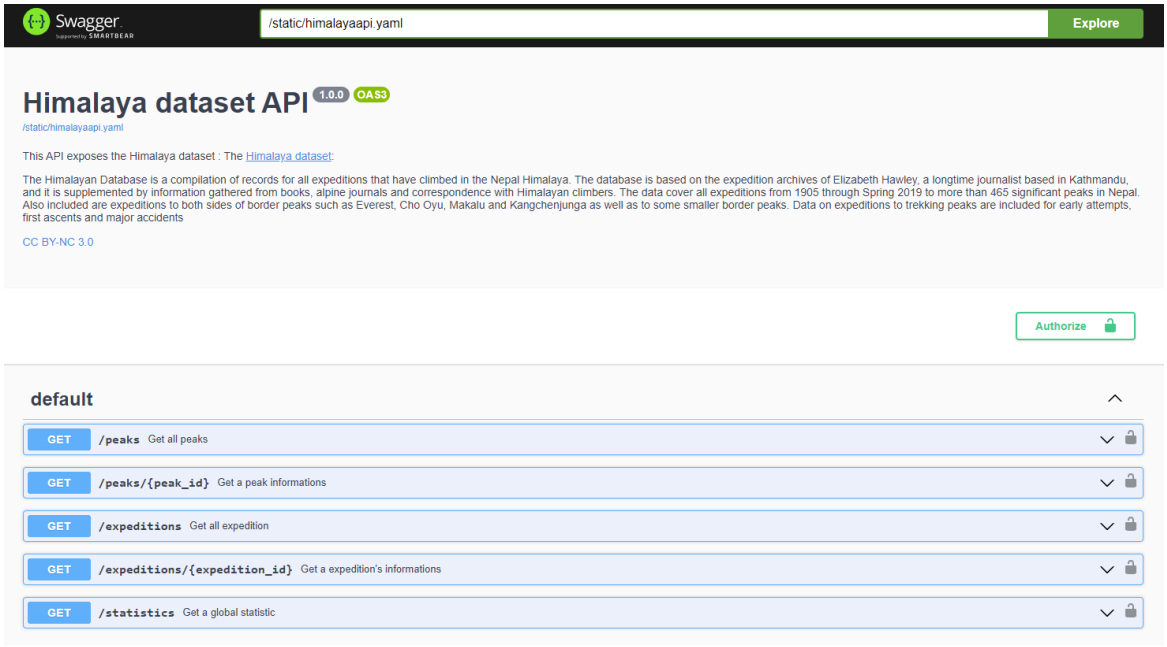
```sql
WITH member_counts AS (
    SELECT
        DISTINCT year AS year,
        COUNT(DISTINCT member_id) AS nb_member
    FROM `da-bootcamp-2023.nivault.members`
    GROUP BY year
),
member_counts_with_lag AS (
    SELECT
        year,
        nb_member,
        LAG(nb_member) OVER (ORDER BY year) AS prev_nb_member
    FROM member_counts
)
SELECT
    year,
    nb_member,
    ROUND((nb_member - prev_nb_member) / prev_nb_member * 100, 2) AS evol_vs_y1
FROM member_counts_with_lag
ORDER BY year;
```

Row	year	nb_member	evol_vs_y1
73	2000	1898	36.74
74	2001	1613	-15.02
75	2002	1578	-2.17
76	2003	2005	27.06
77	2004	2004	-0.05
78	2005	2025	1.05
79	2006	2498	23.36
80	2007	2438	-2.4
81	2008	2579	5.78

11- Exposing Data via API

In order to expose portion of data from the database I have created an API that allows users, such as trekking agencies, to retrieve specific data on expeditions and peaks.
The API is built using Flask with an authentication requirement.
It supports GET requests, with pagination and parameters like, height_min and year.
The API provides responses in JSON format.
http://localhost:8080/

Documentation was made with .yaml file on swagger (/docs) :



Example of code from python file :

```python
@app.route("/peaks")
@auth.required
def peaks():
    #URL Parameters
    page = int(request.args.get('page', 0))
    page_size = int(request.args.get('page_size', MAX_PAGE_SIZE))
    page_size = min(page_size, MAX_PAGE_SIZE)
    include_details = bool(int(request.args.get('include_details',0)))
    height_min = request.args.get('height_min', 0)


    db_conn = pymysql.connect(host="localhost", user="root",password= os.getenv('my_SQL_pw'), database="himalaya",
                              cursorclass=pymysql.cursors.DictCursor)
    # Get the peaks
    with db_conn.cursor() as cursor:
        query = """
                SELECT
                    peak_name,
                    peak_id,
                    height_metres
                FROM peaks
            """
        if height_min is not None:
                query += " WHERE height_metres >= %s"
                params = (height_min, page_size, page * page_size)
        else:
                params = (page_size, page * page_size)
```

Example of the API result for GET/peaks?include_details=1&height_min=8000 :


```
{
  "last_page": "/peaks?page=1&page_size=100&include_details=1&height_min=8000",
  "next_page": "/peaks?page=1&page_size=100&include_details=1&height_min=8000",
  "peaks": [
    {
      "first_ascent_country": "France",
      "first_ascent_year": 1950.0,
      "height_metres": 8091,
      "peak_alternative_name": "Unknown",
      "peak_id": "ANN1",
      "peak_name": "Annapurna I"
    },
    {
      "first_ascent_country": "Spain",
      "first_ascent_year": 1974.0,
      "height_metres": 8026,
      "peak_alternative_name": "Unknown",
      "peak_id": "ANNE",
      "peak_name": "Annapurna I East"
    },
    {
      "first_ascent_country": "W Germany",
      "first_ascent_year": 1980.0,
      "height_metres": 8051,
      "peak_alternative_name": "Unknown",
      "peak_id": "ANNM",
      "peak_name": "Annapurna I Middle"
    },
```


Example of the API result for GET/expeditions/AMAD00102:


```
{
  "basecamp_date": "Sun, 23 Apr 2000 00:00:00 GMT",
  "expedition_id": "AMAD00102",
  "harmonized_agency_name": "Unknown",
  "highpoint_date": "Fri, 05 May 2000 00:00:00 GMT",
  "highpoint_metres": 6814.0,
  "hired_staff": 2,
  "hired_staff_deaths": 0,
  "is_success": 1,
  "member_deaths": 0,
  "members": 5,
  "oxygen_used": 0,
  "peak_id": "AMAD",
  "peak_name": "Ama Dablam",
  "season": "Spring",
  "termination_date": "Tue, 09 May 2000 00:00:00 GMT",
  "termination_reason": "Success (main peak)",
  "trekking_agency": "Unknown",
  "year": 2000
}
```

12- Conclusion

In conclusion, the in-depth analysis of Himalayan expeditions and the development of predictive tools prove crucial for a more sustainable and secure management of high-altitude mountaineering. The figures are eloquent: in 50 years, the number of climbers has increased tenfold, multiplying risk factors on the high peaks. Concurrently, the widespread use of oxygen, up by 1077% over the same period, has made the inaccessible accessible, radically transforming the nature of these expeditions.

Faced with this evolution, industry professionals are confronted with a major, multidimensional challenge: maintaining the safety of expeditions and personnel while preserving the environmental integrity of these unique spaces. This project, by equipping the Union of Himalayan Agencies with data-driven insights and advanced prediction capabilities, will make a significant contribution to addressing these challenges of overcrowding and safety.

The ultimate goal transcends mere logistical management: it's about preserving the authentic and transformative experience of Himalayan mountaineering, while minimizing its ecological footprint and maximizing climber safety.

This holistic approach, balancing adventure, safety, and preservation, paves the way for a promising future. It will maintain the accessibility of the world's highest peaks, but in a responsible and sustainable manner, ensuring that future generations can also experience this extraordinary adventure under optimal conditions.

13- GDPR

After a thorough evaluation of the data collected for this project, I can confidently say that no personal data was used in any part of the processes. Data sources are entirely public.
The 'members' table only contains member_id and not names.
Therefore, this project adheres to the guidelines and principles of General Data Protection Regulation (GDPR).

14- References

- Trello :
https://trello.com/b/wkjmP5si/final-project

- Kaggle dataset :
https://www.kaggle.com/datasets/majunbajun/himalayan-climbing-expeditions?select=peaks.csv

- API used :
https://nominatim.org/release-docs/latest/api/Overview/
https://openweathermap.org/history (⇒ failed)

- Web scraping:
https://www.topchinatravel.com/mount-everest/the-climate-of-mount-everest.htm

- Tableau public :
https://public.tableau.com/views/Himalaya-expeditions-Story/Histoire1?:language=fr-FR&:sid=&:redirect=auth&:display_count=n&:origin=viz_share_link