Data Analytics

Visually Similar Pictures Recommender applied to Travel Industry

“What if I could choose my next destination based on a picture I like.”

Final Project, March 2023, delivered by:

Kevin THAI

Table of Contents

Introduction

I believe that successful companies are those who can leverage technology in order to push the boundaries of tailored customer experiences. These companies or service-providers gain significant competitive advantages by always understanding their customers' changing needs while being able to simplify, enhance and entertain them during the purchase decision journey.

To illustrate such "Customer First" vision, we could think about E-commerce's success and how they leverage purchase history and browsing behavior data to provide personalized product recommendations. No doubt that these algorithms helped boosting basket value while saving customers' time! We could also think about how fashion or beauty companies had to re-invent themselves (especially during sanitary restrictions times) to propose virtual try-on experiences for online shoppers leveraging the power of computer vision and augmented reality. Finally, grocery stores without checkout lines or cashiers did make newspaper headlines when they were launched!

Did you ever wonder how e-commerce websites would predict your next purchase or how a computer could recognize an object? As far as I am concerned, I was always intrigued and curious about the technology, data and machine learning techniques behind these enhanced customers experiences. So, I thought that I could explore and learn more on this topic during my last project at Ironhack by picking a project that will mix data collection and analysis, image recognition and clustering techniques.

My final project is called "**Visually Similar Travel Pictures Recommender**". Here is the pitch: tired of visiting the same popular places and willing to dream further? The promise is simple: the user inputs a travel picture that inspires her/him (ex: beautiful landscape) and the machine returns a couple of similar pictures around the world (hopefully)! Isn't it exciting?

This project consists in two main parts:

1. Leverage Pexels API (Pexels is a website that offers a vast collection of free high-quality photos, videos and illustrations) to gather more than 3,900 pictures from top 50 most visited countries in the world associated with pictures features (ex: dimension, photographer, description) to generate preliminary insights.
2. Apply machine learning techniques to cluster these pictures based on image content similarities and suggest similar pictures based on user's input.

The purpose of this document is to walk you through the first part of this project (from data collection, cleaning to exploratory data analysis with SQL and Python). The second part is not covered in this document and will be showcased during the presentation!

Ladies and gentlemen, please fasten your seatbelts, we are preparing for take-off!

Plan

The goal of this section is to provide an overview of this project's key steps. It will hopefully help the reader to get a better understanding of following sections.

1. **Data Collection:**
    a. Small database from Kaggle, listing Top Visited Countries (49 in total) in the world in 2021.
    b. Pexels API request to collect 80 pictures per country resulting in a dataset of ~3,900 rows and 8 columns (ex: pictures_id, dimension and description).

2. **Data Cleaning:**
    a. Mostly dealing with description (string) cleaning to remove special characters and standardize in lowercase in anticipation of NLP/word clouds in exploratory data analysis.
    b. Preparing all the tables/entities in Python and exporting to MySQL.

3. **Exporting to MySQL database:**
    a. Why did I choose SQL?
    b. Building relationships between entities (ERD attached).

4. **Exploratory Data Analysis:**
    a. Explore the data with 5 SQL queries.
    b. For each query, follow-up on Python deep-dive to explore more with Data Visualization.

5. **Machine Learning (not covered in this document)**

Data Collection

1. Why did I choose Pexels API to collect my data?

I chose Pixels API to build my pictures dataset for two main reasons:

1. Unlike other APIs, Pexels API is very convenient: it enables up to 200 requests per hour which is by far more than enough to build my dataset. Such “high rate-limit” gave me the possibility to do some tests and trials and to make mistakes without being “punished” and having to wait another day to pull more data.
2. Since, I’m working on image recognition applied to Travel industry, I wanted to find high-quality and inspiring pictures. As you can see below, I found that Pexels top search results were appealing!

2. How did I proceed to collect my data?

I first use a dataset found on Kaggle containing the 49 most visited countries in the world in 2021. As you can see below, I build a “Country_travels” column and store all the values into a list. Later, I will iterate through this list to query Pexels API with “France travels” then “Spain travels” etc. as query search keywords (in the requests parameters). Note: I have added the keyword “travels” to improve the top search pictures results.

Country_id	Country_name	touristArrivals	Country_travels
1	France	89400000	France travels
2	Spain	83700000	Spain travels
3	United States	79300000	United States travels
4	China	65700000	China travels
5	Italy	64500000	Italy travels

```
liste
```

```
['France travels',
 'Spain travels',
 'United States travels',
 'China travels',
 'Italy travels',
 'Turkey travels',
```

a. JSON files bulk extraction using a loop.

First, I retrieved the data for only one country (France) to make sure I understand how did the API work and how does the data look like. Then, I decided to iterate through my previous list of most visited countries in the world defining the below function in Python:

```python
# Let's define a function to extract our data from Pexels API and store in "all_results" (aggreg. of all JSON files)

def api_extract(pictures_count,nb_page):

    url = 'https://api.pexels.com/v1/search'
    # list that we have defined in the first part, "country + travels"
    countries = liste

    # Create an empty list to store all the search results
    all_results = []

    # Loop through the list of countries and retrieve photos for each country
    for country in countries:
        # Set up the request headers with my API key
        headers = {'Authorization': API_KEY}

        # Set up the request parameters with the search keyword and the number of results to return
        params = {'query': country, 'per_page': pictures_count, 'page': nb_page}

        # Send the API request
        response = requests.get(url, headers=headers, params=params)

        # Parse the response JSON to retrieve the search results
        results = response.json()['photos']

        # Append the search results to the all_results list
        all_results.extend(results)

        # Pause for a few seconds between requests to avoid hitting the API limit or being called out
        time.sleep(3)

    return all_results
```

It's worth noticing that I used the time.sleep(3) options within the loop so that I reduced the risk of being banned (even though the risk was very low). Then, I ran my function with following inputs: 80 (which corresponds to the maximum number of pictures I can retrieve per page) and 1 (meaning first page results). After obtaining the aggregated JSON, I decided to save it on my local folder as a first "checkpoint".

At this stage, I have a JSON aggregated files (a list of dictionaries) that gathers 80 pictures' information per country so roughy 3,900 rows. Now, I need to format this file into a dataframe while selecting the columns I would like to keep.

b. Turning the JSON files into a dataframe with selected columns

From the JSON files, I decided to keep the following columns to continue with my analysis:

```
['Pictures_id', 'Width', 'Height', 'Full URL', 'Photographer_id','Photographer', 'Avg Color', 'Description', 'Country_id']
```

While the columns' content is quite straightforward, please note that Full URL refers to the picture's URL, Avg Color refers to the dominant color of the picture (Hex format) and Country_id is an auto-incremental index (starting from 1 and increasing every 80 rows).

```python
def create_df():
    # Create the frame with columns name
    df = pd.DataFrame(columns=['Pictures_id', 'Width', 'Height', 'Full URL', 'Photographer_id','Photographer', 'Avg Color', 'Description', 'Country_id'])
    for i, result in enumerate(all_results):
        # Make sure it matches with above for columns name
        row = pd.DataFrame({
                'Pictures_id': result['id'],
                'Width': result['width'],
                'Height': result['height'],
                #'Tiny URL': result['src']['tiny'],
                'Full URL': result['url'],
                'Photographer_id': result['photographer_id'],
                'Photographer': result['photographer'],
                'Avg Color': result['avg_color'],
                'Description': result['alt'],
                'Likes': result['liked'],
                'Country_id': (i // 80) + 1 # increment every 80 rows
            }, index=[0])
        # iterating and building back the country_id
        df = pd.concat([df,row], ignore_index=True)
    return df
```

After calling the function, I obtain my dataframe which looks like:

Pictures_id	Width	Height	Full URL	Photographer_id	Photographer	Avg Color	Description	Country_id	Likes
13918727	5304	7952	https://www.pexels.com/photo/sculpture-and-streetlight-near-eiffel-tower-in-paris-france-13918727/	5767088	Eugenia Remark	#5E5141	Black Statue of Man on Top of Building	1	False
14918481	2000	2500	https://www.pexels.com/photo/photo-of-a-mont-san-michele-abbey-in-france-14918481/	406314056	paul jousseau	#6C635C	Photo of a Mont San Michele Abbey in France	1	False

Data Cleaning

a. Checking null values, removing duplicates and string standardization

Given that I built my dataset from Pexels API (high-quality data), I did not expect any null values. But since I selected the top 80 pictures appearing on the top search per country, I could have some duplicated pictures hence the use of drop_duplicates. Anticipating some NLP work for my data visualization part, I decided to standardize all columns containing string by removing special characters and transform in lower case.

```
def df_cleaning(df):
    # even though I don't have null or duplicates
    cleaned_df = df.drop_duplicates(subset = "Pictures_id").dropna()
    # anticipating SQL loading or text processing later, I prefer to remove special characters for desc and photographer name
    special_chars = {
        "[éèêë]": "e",
        "[áàâäã]": "a",
        "[óòôöõ]": "o",
        "[úùûü]": "u",
        "[íìîï]": "i",
        "[ñ]": "n",
        "[ç]": "c",
        "[ß]": "ss"
    }
    for pattern, replacement in special_chars.items():
        cleaned_df['Photographer'] = cleaned_df['Photographer'].apply(lambda x: re.sub(pattern, replacement, str(x), flags=re.IGNORECASE))
        cleaned_df['Photographer'] = cleaned_df['Photographer'].str.lower()  # convert to lowercase
        cleaned_df['Description'] = cleaned_df['Description'].apply(lambda x: re.sub(pattern, replacement, str(x), flags=re.IGNORECASE))
        cleaned_df['Description'] = cleaned_df['Description'].str.lower()  # convert to lowercase
    return cleaned_df
```

b. Preparing tables to export in MySQL database.

Below I will give an overview of what the tables/entities look like. If you remember well, 2 tables were obtained from Kaggle (most visited countries) and from Pexels API (pictures information). Additionally, I have created 2 mores tables/entities from the main table (Pexels API): 1) Photographer Mapping table containing unique values for photographer_id and photographer (name) and 2) Color mapping table obtained by converting Hex column into RGB components. Let's have a quick look:

1. Countries_mapping (shape: 49x5):

Country_id	Country_name	touristArrivals	region	sub-region
1	France	89400000	Europe	Western Europe
2	Spain	83700000	Europe	Southern Europe
3	United States	79300000	Americas	Northern America

2. API_data (shape: 3682x8):

Pictures_id	Width	Height	Full URL	Photographer_id	Avg Color	Description	Country_id
13918727	5304	7952	https://www.pexels.com/photo/sculpture-and-streetlight-near-eiffel-tower-in-paris-france-13918727/	5767088	#5E5141	black statue of man on top of building	1
14918481	2000	2500	https://www.pexels.com/photo/photo-of-a-mont-san-michele-abbey-in-france-14918481/	406314056	#6C635C	photo of a mont san michele abbey in france	1

3. Photographers_mapping (shape: 1515x2) and 4. Colors_mapping (shape: 3618x4).

Photographer_id	Photographer
5767088	eugenia remark
406314056	paul jousseau
956039	jill evans

Hex	R	G	B
#5E5141	94	81	65
#6C635C	108	99	92
#7B7B79	123	123	121

Note: Colors_mapping table was obtained by converting the Hex value into its RGB components using:

```
# I discover that we could turn a HEX to RGB using the "hexadecimal"  applied to
# define a function to convert hexadecimal color codes to RGB values
def hex_to_rgb(hex_code):
    hex_code = hex_code.lstrip('#')
    return tuple(int(hex_code[i:i+2], 16) for i in (0, 2, 4))

# apply the 'hex_to_rgb' function to the 'Hex' column of the DataFrame
unique_colors_df[['R', 'G', 'B']] = pd.DataFrame(unique_colors_df['Hex'].apply(hex_to_rgb).tolist())
```

Now these 4 entities are ready to be exported in MySQL database!

Exporting to MySQL database

a. Why did I choose SQL?

I chose SQL for several reasons:

- Great at handling structured data that is organized into tables with defined columns and rows. It's important in data analysis to be able to store and retrieve data in a consistent way (string, integer, date).
- Powerful analytics capabilities: SQL offers a wide range of analytical functions that allow data analysts to perform calculations, grouping and filtering.
- It's widely used by companies of all sizes. Many companies use SQL databases to store and manage their data, and SQL is the standard language used for querying and analyzing data in these databases. So, it's important to be proficient in this language since it's an asset for job hunting!
- We learnt it during the bootcamp, and I also have previous experiences in SQL.

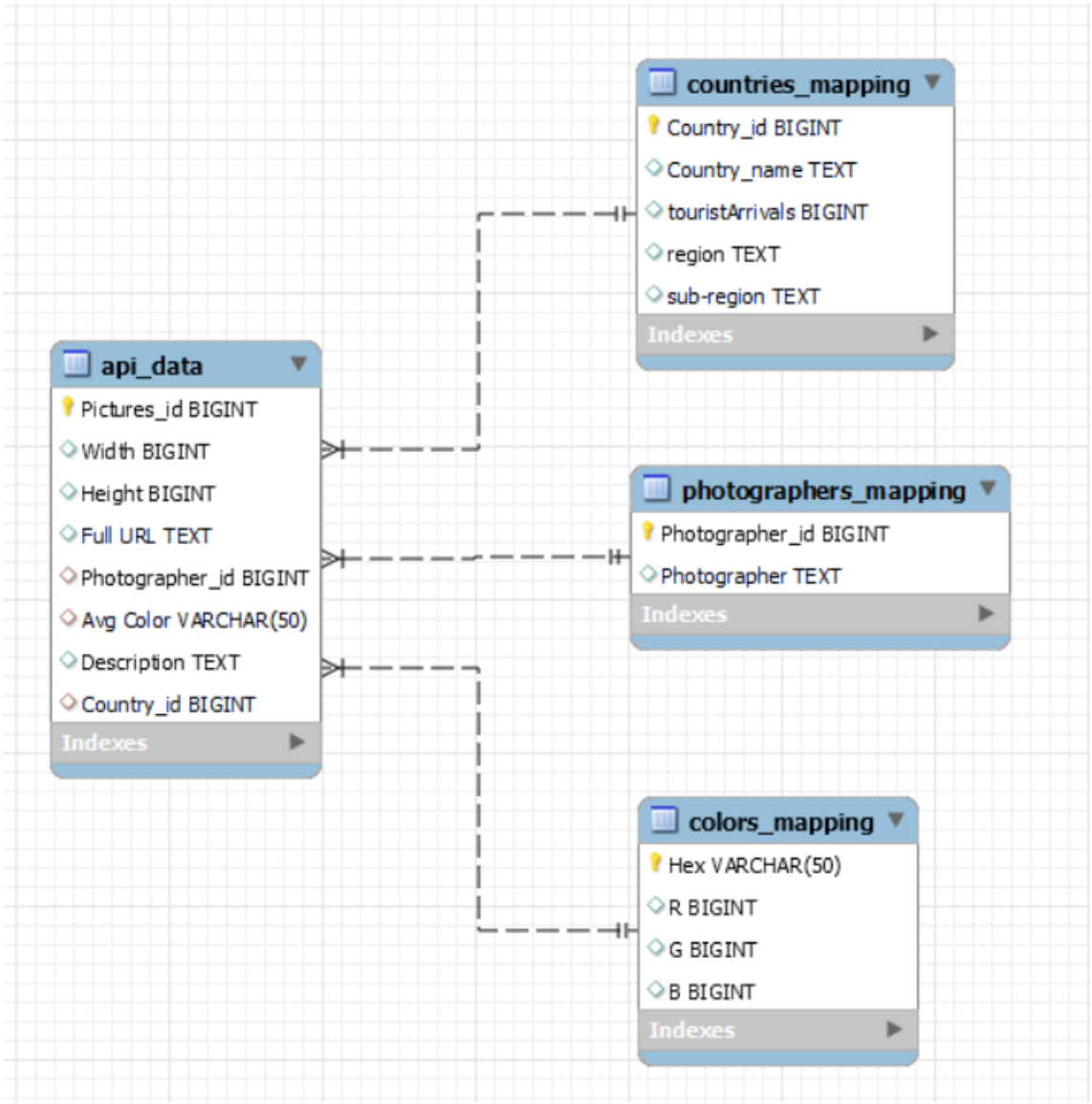
b. From Python to MySQL database (ERD attached)

Now that we have our 4 tables/entities (see above), I created a database in MySQL and used sqlalchemy in Python to export my entities into MySQL. As a next step, I defined the relationships between tables by setting up primary keys and foreign keys as below:

```sql
-- Creating DB schema so that I can import from my jupyter notebooks
CREATE DATABASE Final_Project;
use Final_Project;

-- Modifying some columns type for setting up keys
ALTER TABLE colors_mapping MODIFY Hex VARCHAR(50);
ALTER TABLE api_data MODIFY `Avg Color` VARCHAR(50);

-- Setting Primary & Foreign Keys (FK are in the main table and are joined to smaller/mapping tables PK)
ALTER TABLE colors_mapping ADD PRIMARY KEY (Hex),
ALTER TABLE api_data ADD FOREIGN KEY (`Avg Color`) REFERENCES colors_mapping(Hex);

ALTER TABLE countries_mapping ADD PRIMARY KEY (Country_id);
ALTER TABLE api_data ADD FOREIGN KEY (Country_id) REFERENCES countries_mapping(Country_id);

ALTER TABLE photographers_mapping ADD PRIMARY KEY (Photographer_id);
ALTER TABLE api_data ADD FOREIGN KEY (Photographer_id) REFERENCES photographers_mapping(Photographer_id);

ALTER TABLE api_data ADD PRIMARY KEY (Pictures_id);
```

After setting up the relationships between entities, I end up having this ERD:

How to read: ONE country can be associated to MANY pictures but ONE picture is associated to ONE country. Same applies for photographer and dominant color.



Now we are ready for the EDA/SQL queries part! Let's explore!

Exploratory Data Analysis using SQL and Python

In below section, we will explore our data leveraging both SQL and Python to generate insights. I divided this section into “insights themes” such as: photographers and pictures breakdown, pictures’ dimension, pictures’ dominant color and natural language processing on pictures’ description. Except for NLP, I will introduce each theme with a SQL query (5 queries in total) and follow-up with a Python deep-dive and data visualization.

Please note that the analysis performed below is based on a sample of ~3,900 pictures on Pexels and can not be generalized to the whole website’s content.

a. Photographers and pictures breakdown

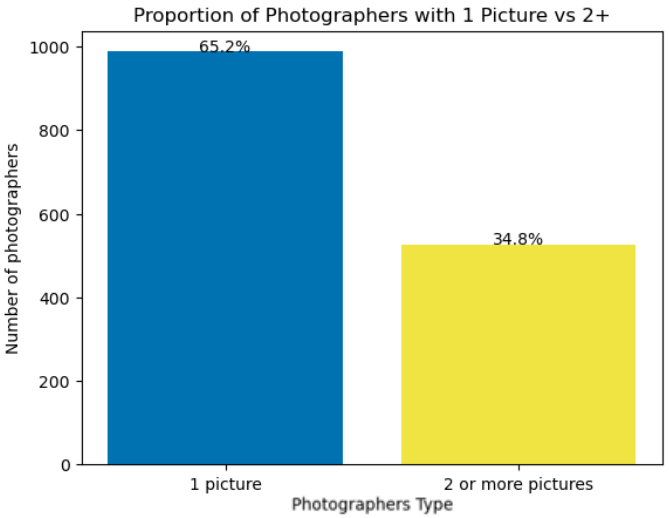
First, we begin our data exploratory journey by answering to the following questions: who are the top photographers in terms of pictures’ contribution (measured in count)? What is the split between those who published only one picture and those who published 2+? And finally, are they “specialized” in one country or do they tend to publish pictures from several countries (for photographers who published at least 2 pictures)?

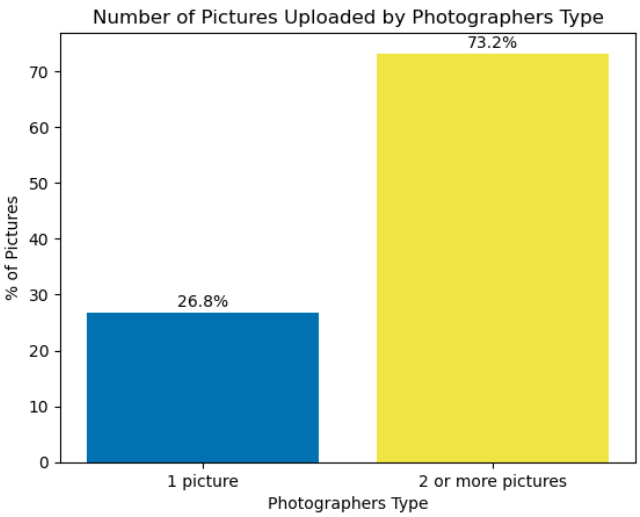
```sql
-- Query 1 : Who are the top contributors in terms of pictures ?

SELECT
b.Photographer_id,
b.Photographer,
COUNT(a.Pictures_id) AS pictures_count
FROM api_data a
JOIN photographers_mapping b ON a.Photographer_id = b.Photographer_id
GROUP BY
b.Photographer_id,
b.Photographer
ORDER BY pictures_count DESC
LIMIT 10
;
```

Result Grid | Filter Rows:

Photographer_id	Photographer	pictures_count
2659	pixabay	203
1437723	cottonbro studio	47
1522664	taryn elliott	46
2272619	rachel claire	42
102775	magda ehlers	42

This first query gives us a sense of the top 10 contributors in terms of pictures count. We can see that pictures could come from several sources such as existing websites (ex: pixabay), professional studio or individuals.

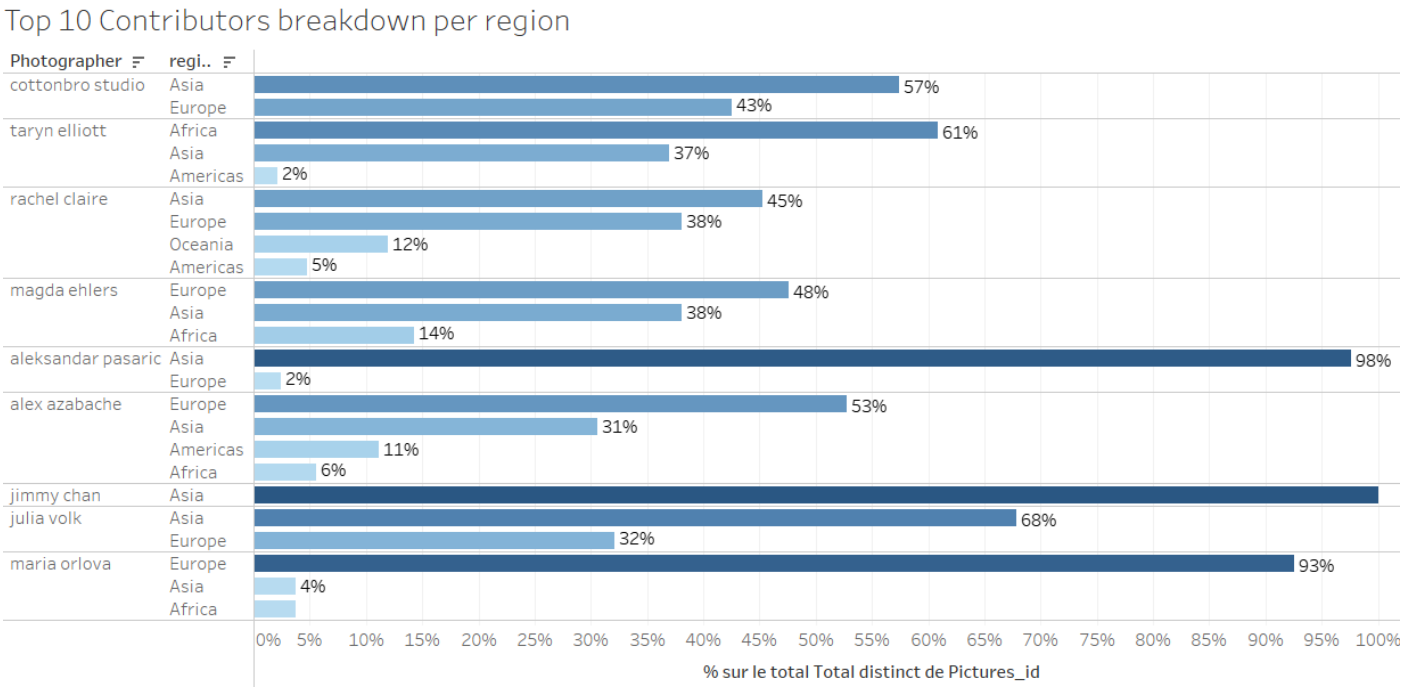




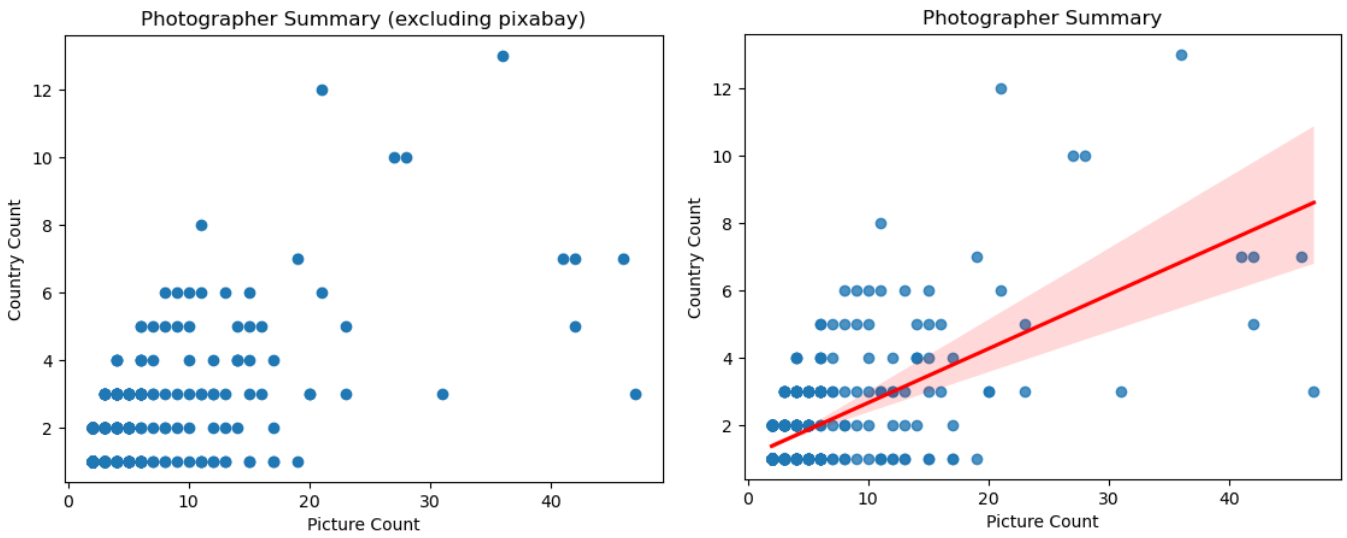
Note: in our case, we are not necessarily interested in excluding outliers but pixabay could be considered as one. Let's further deep dive in Python!

From previous charts, we can see that in our sample, 65% of photographers only published one picture accounting for 27% of total pictures count (988) meaning that photographers who published at least 2 pictures accounted for 73% of total pictures! Now, I am curious to see if those who published at least 2 pictures are "specialized" in one country or tend to publish pictures from different countries.

Looking at the chart below for the top 10 contributors (removing pixabay), it seems like they are mostly covering several regions (therefore countries). For instance, cottonbro studio covered both Asia and Europe regions.



Checking back on Python (using scatterplot and regplot): when we only look at photographers who published at least 2 pictures, we can say there is positive correlation between pictures count and country count: the more they publish, the more likely they are to produce diverse content (several countries).



To summarize, we learn that in our sample, most of the photographers published only one picture. But more than 7 out of 10 of the pictures are published by photographers who published at least 2 pictures. These photographers tend to propose diverse content (covering more than one country). But what do they publish?

b. Pictures' dimension

In this section, I will have a look on our sample's pictures' dimension. Do photographers' upload small, medium or large pictures? Do they have any preferences for landscape or portrait orientation? Let's first ask SQL!

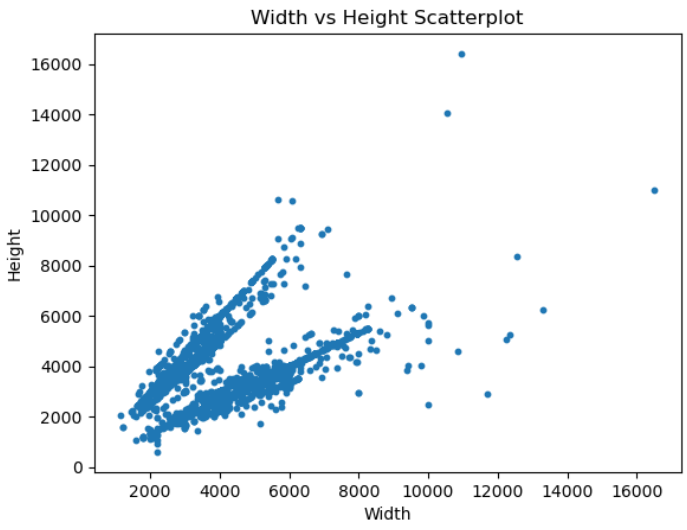
```sql
-- Query 2 : What are the average dimension of pictures published by region in the world ?

SELECT
b.region,
AVG(a.Width) AS average_width,
AVG(a.Height) AS average_height
FROM api_data a
JOIN countries_mapping b ON a.Country_id = b.Country_id
GROUP BY b.region
;
```

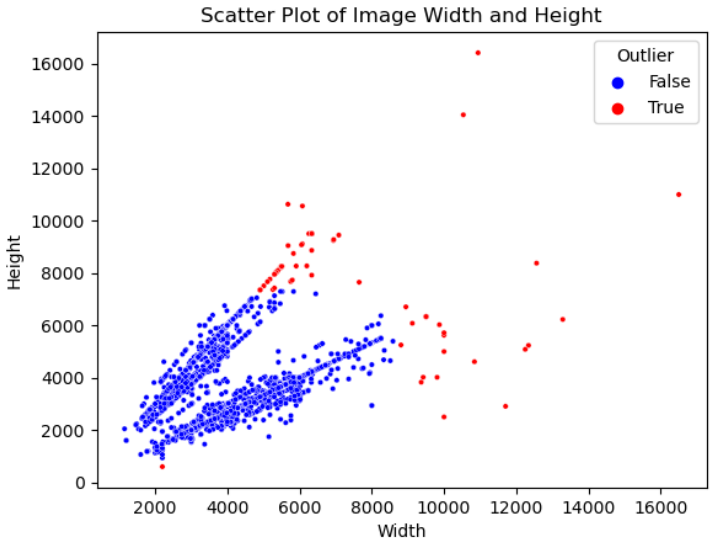
region	average_width	average_height
Europe	4361.5572	4026.4104
Americas	4261.9245	4162.7987
Asia	4452.9935	4020.9416
Africa	4299.0580	3997.3823
Oceania	4358.2208	3879.8701

```sql
-- Query 3 : From previous, in avg, pictures look really "big", let's deep dive a bit using CASE WHEN

WITH subquery AS(
SELECT
   Pictures_id,
   CASE WHEN Width <= 500 AND Height <= 500 THEN 'small'
        WHEN (Width >= 500 AND Width <= 1200) AND (Height >= 500 AND Height <= 1200) THEN 'medium'
        WHEN (Width > 1200) AND (Height >= 1200) THEN 'large'
        ELSE 'unclassified'
   END AS picture_category
FROM api_data
)
SELECT
b.picture_category,
COUNT(a.Pictures_id)
FROM api_data a JOIN subquery b ON a.Pictures_id = b.Pictures_id
GROUP BY
b.picture_category
```

picture_category	COUNT(a.Pictures_id)
unclassified	13
large	3669

Wow, it's impressive but it seems like our dataset mostly contains large pictures with average width and height above 4,000 pixels. Stepping back this is not so surprising since Pexels is known for providing high-quality pictures. Let's explore now on Python: the scatterplots below confirm our assumption that our dataset mostly contains large pictures (>1200 pixels for width and height). Note: while we don't exclude outliers in our project, it's worth mentioning that applying IQR method on both columns, we would have excluded red data points being pictures with too high or low width/height.





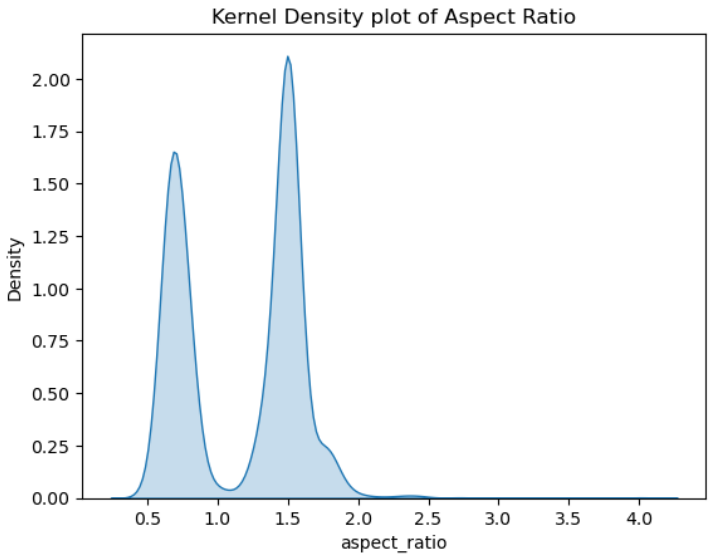
Having a closer look on our charts, it looks like we have a “two axis” separation in our data points distribution. Could it be linked to aspect ratio (defined as width/height) meaning we have landscape and portrait orientation as predominant pictures’ format? Let’s ask SQL!

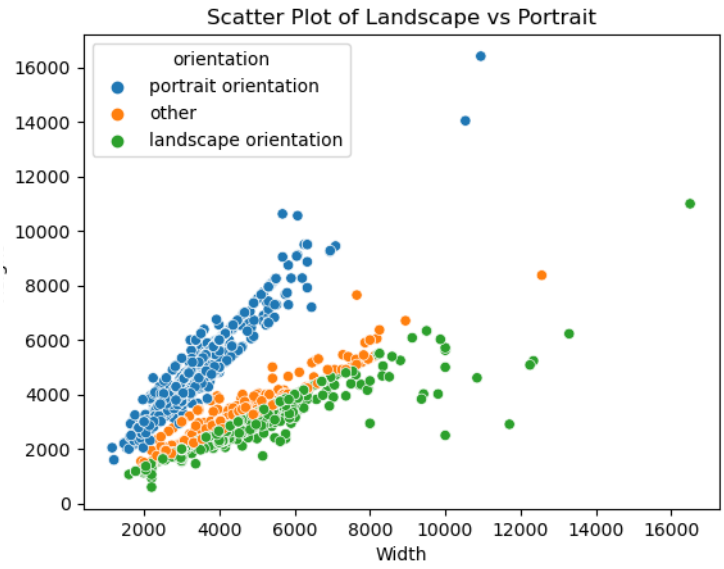
```sql
-- Query 4: Pretty obvious, let's see if we have landscape images ?
WITH subquery AS(
SELECT
   Pictures_id,
   CASE WHEN Width / Height >= 1.5 THEN 'landscape'
        ELSE 'not_landscape'
   END AS picture_category
FROM api_data
)
SELECT
b.picture_category,
COUNT(a.Pictures_id)
FROM api_data a JOIN subquery b ON a.Pictures_id = b.Pictures_id
GROUP BY
b.picture_category
;
```

picture_category	COUNT(a.Pictures_id)
not_landscape	2216
landscape	1466

It looks like that we had a good intuition on this “two axis” separation. It’s linked to the landscape/portrait orientation (knowing that portrait orientation is defined as aspect ratio >= 1.5). Let’s go back to Python and visualize the previous chart differently adding orientation as “hue”.

In the left chart below, we have plotted the probability density function of the aspect ratio, we can see to peaks centered around 0.75 and 1.5 which respectively correspond to the definition of “portrait” and “landscape” pictures. Another way of visualizing this breakdown is on the right chart below: we have 1,559 portrait pictures, 1466 landscape pictures and 657 pictures close to a square.





What did we learn? In this part, we learn that Pexels does comply with its promise of offering free high-quality pictures since most of the pictures of our sample are considered as large. Also, we can see that the most popular orientations in photography (portrait and landscape) are represented in our dataset.

Now that we have explored pictures’ dimension, what about exploring the main colors?

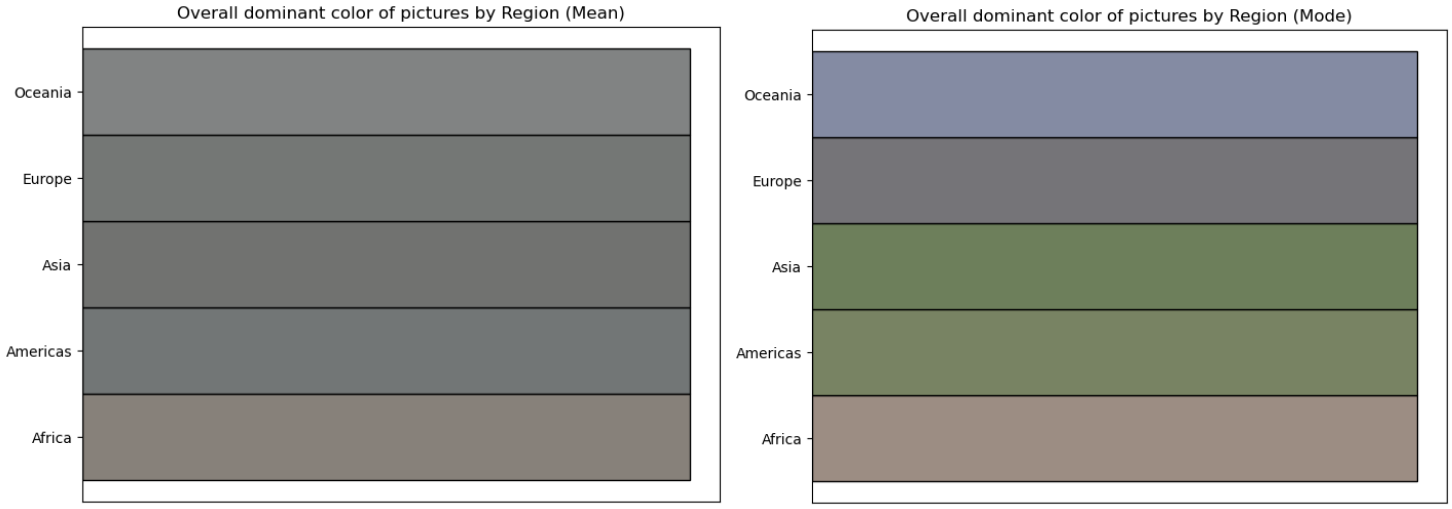
c. Pictures' colors

Remember about our Hex column (representing the dominant color of the picture according to Pexels API) that I also converted into RGB (Red, Green, Blue) components? Let's have a further look into this with SQL!

```sql
-- Query 5: RGB
SELECT
c.region,
ROUND(AVG(b.R),0) AS avg_red,
ROUND(AVG(b.G),0) AS avg_green,
ROUND(AVG(b.B),0) AS avg_blue
FROM api_data a
JOIN colors_mapping b ON a.`Avg Color` = b.Hex
JOIN countries_mapping c ON a.Country_id = c.Country_id
GROUP BY
c.region
```

region	avg_red	avg_green	avg_blue
Asia	113	114	112
Europe	116	119	117
Oceania	129	131	131
Africa	135	129	122
Americas	114	118	118

In the query above, I retrieved the average component (RGB) per region. The idea behind is to "plot" these colors so that we can visually see if the colors' tone are different across regions in the world (which should be the case). Let's now switch into Python and plot these colors (creating bar charts with homogeneous size with colors being the average red, green and blue values):



On the charts above, I have plotted the dominant colors per region using 1) the average RGB of each region (output of SQL, left side chart) and 2) using the mode of RGB components per region (right side chart). We can visually conclude that the average method was the most intuitive but not the most effective. It looks like average method tend to "darken" the pictures' colors' tone with higher values whereas the mode method provided more conclusive results.

The dominant colors displayed in the right chart can be interpreted instinctively: Oceania region tends to be blue (most likely due to sea), Europe tends to be greyer (Eifel tower impact?), Asia and Americas turn to be greener (maybe due to natural park or rice field) and finally Africa is more "sandy" (probably due to desert and pyramid). Isn't it interesting? (at least it gives us confidence in the data provided by the API!).

Can you guess? Does it match with the insights above using mode?

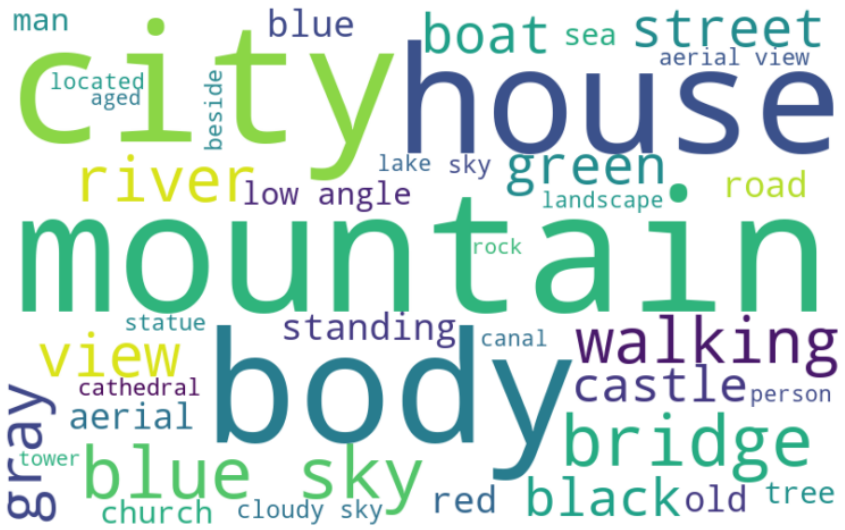
In this part, we learnt that the mode (on RGB) was the most effective way to visualize the most frequent dominant colors' tone per region. This simple but powerful insights gave us confidence in Pexels API data's reliability.

Ready for the last part? Let's go exploring the sky with word clouds!

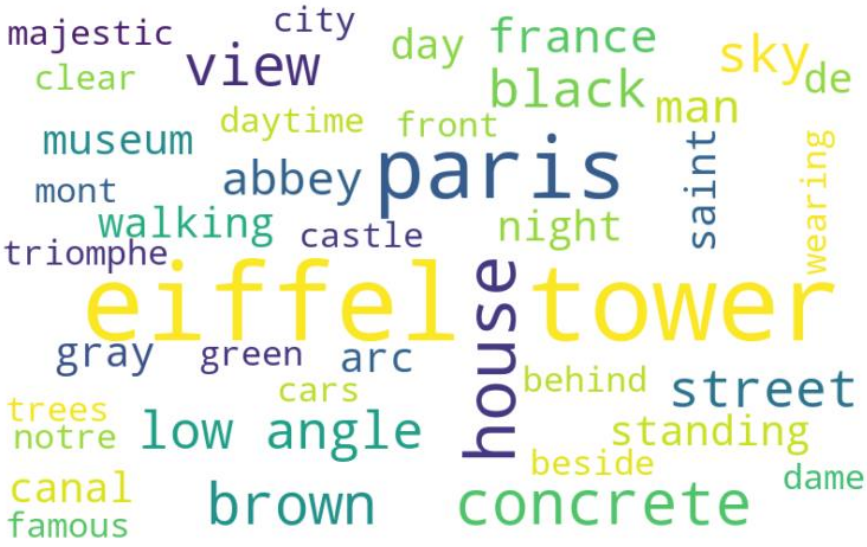
d. NLP: word clouds with pictures' description

Finally, NLP and word clouds based on most frequent words used in pictures' description will be our last stop in this thrilling data exploratory journey! Note that before performing world clouds in Python: we have standardized strings (lowercase, removing special characters), converted into strings and removed a list of stopwords (using the python library and some curated words).

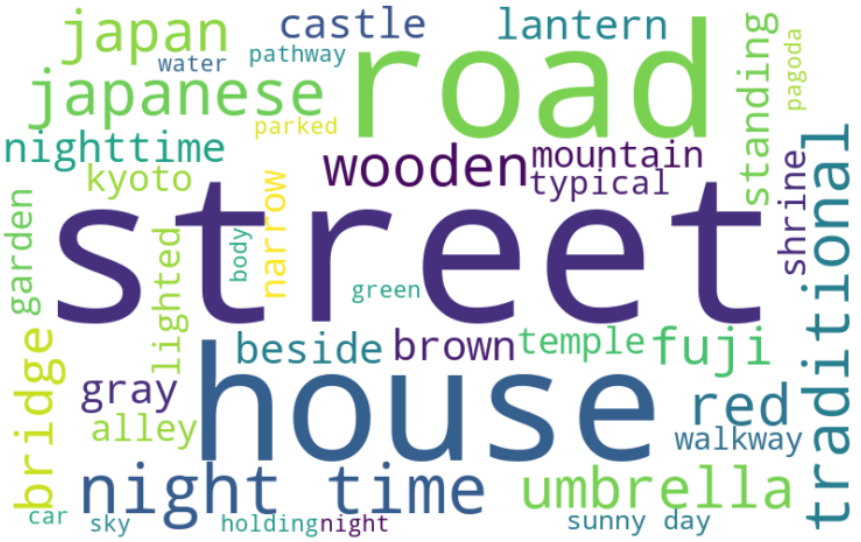
Let's start with Europe, as you can see below, "city, house, mountain" are among the most frequent words along with "walking, castle or bridge" depicting modern infrastructures and historical vestiges!



Now, if we go one step down, at country level and selecting France, we obtain the words cloud below. It's amazing to capture more specific words such as "Eiffel tower, notre dame, abbey" or "majestic" depicting the magic of the City of Lights (at least its touristic side)!



To conclude, we will travel far away and investigate Japan! Let's now conclude our last stop with few relaxing words: "garden, traditional, fuji and lantern". Do you feel relaxed?

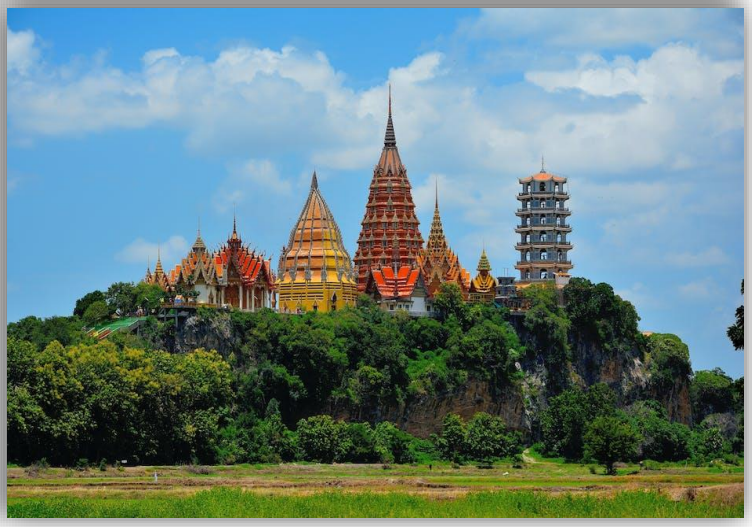


A little surprise before concluding…

Please find below, some pictures I found inspiring from my dataset (Pexels API), enjoy!





Conclusion

Ladies and gentlemen, we are landing at Ironhack Paris airport shortly. Thank you for flying with us today but before taking off your seatbelt, please remember what we did and learnt today:

1. We leveraged Pexels API to build our own dataset composed of ~3,900 rows containing links to top visited countries' pictures (in high-quality) and information about their dimension, dominant colors or description.
2. After cleaning our dataset, we have exported 4 tables/entities into MySQL database and performed exploratory data analysis combining SQL and Python data analysis capabilities.
3. During this thrilling exploratory journey, we have learnt that:
    a. In our sample, most of the photographers only published one picture but those who published at least 2 accounted for more than 73% of total pictures count and tend to cover several countries (diversified)
    b. By far, photographers tend to prefer uploading large pictures (with average width and height above 4,000 pixels) ensuring the best quality for users. Most popular orientations in photography (landscape and portrait) are represented in our dataset in equal proportion.
    c. The most effective way to visualize the most frequent dominant colors' tone per region was to use the RGB components' mode. The colors that we plotted were in line with our expectations and strengthened the confidence we had on Pexels API data's reliability.
    d. NLP on pictures' description enabled us to capture the most frequent words used in our sample such as "mountain, blue sky or city" as well as very specific words at country level such as "notre dame, Eiffel tower or majectic" for France.

Please stay tuned since this is only the beginning of our journey, next step is: cluster our dataset of pictures based on content similarities and be able to suggest new travel destinations to users based on their pictures' preference!