Final Project "Information"

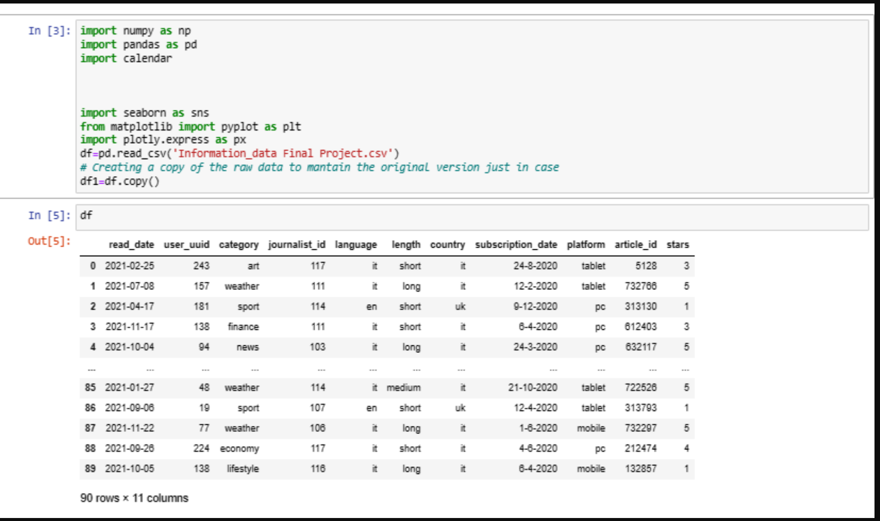
Alessandro Smajlovic

INTRO

i

- As a point of reference we take an online news outlet that creates science and technology-themed content, with the aim of making it easier for the widest possible audience to understand science and technology news.

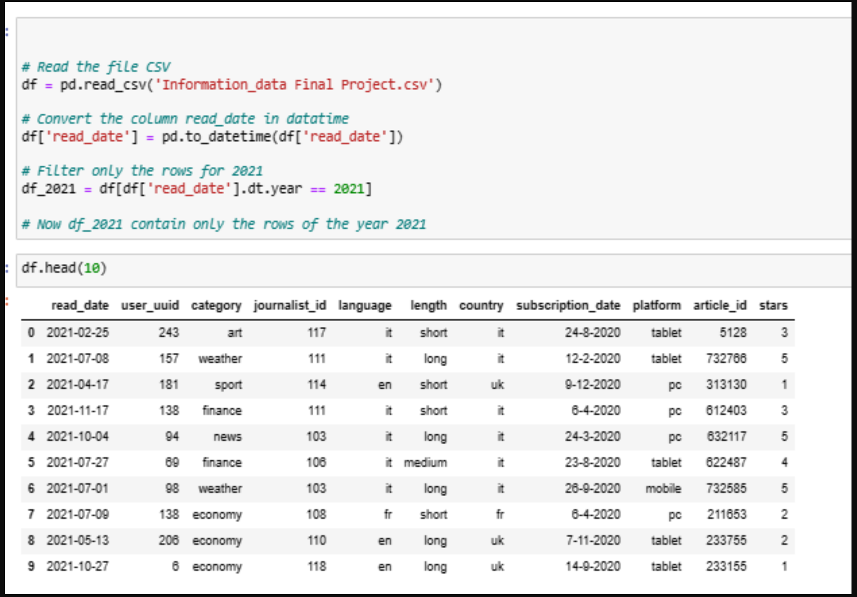
- We decide to analyze the most read articles in 2021.
- The aim of the analysis is to to understand the tastes of the public, what are the Most read articles, most popular content and the trend of registrations. In this regard, a customer segmentation, with the aim of creating a strategy
- For this we are going to do a first skimming and then perform an EDA on Python in which we will analyze the composition of the dataset and of the variables, seeing the trend of the categories in the chosen year.
- At that point, we create dashboards in Tableau to show the division among personas
- Here's the source: Information csv



```
In [3]: import numpy as np
        import pandas as pd
        import calendar



        import seaborn as sns
        from matplotlib import pyplot as plt
        import plotly.express as px
        df=pd.read_csv('Information_data Final Project.csv')
        # Creating a copy of the raw data to mantain the original version just in case
        df1=df.copy()

In [5]: df

Out[5]:
```

	read_date	user_uuid	category	journalist_id	language	length	country	subscription_date	platform	article_id	stars
0	2021-02-25	243	art	117	it	short	it	24-8-2020	tablet	5128	3
1	2021-07-08	157	weather	111	it	long	it	12-2-2020	tablet	732766	5
2	2021-04-17	181	sport	114	en	short	uk	9-12-2020	pc	313130	1
3	2021-11-17	138	finance	111	it	short	it	6-4-2020	pc	612403	3
4	2021-10-04	94	news	103	it	long	it	24-3-2020	pc	632117	5
...	...	...	...	...	...	...	...	...	...	...	...
85	2021-01-27	48	weather	114	it	medium	it	21-10-2020	tablet	722526	5
86	2021-09-06	19	sport	107	en	short	uk	12-4-2020	tablet	313793	1
87	2021-11-22	77	weather	106	it	long	it	1-6-2020	mobile	732297	5
88	2021-09-26	224	economy	117	it	short	it	4-6-2020	pc	212474	4
89	2021-10-05	138	lifestyle	116	it	long	it	6-4-2020	mobile	132857	1

90 rows × 11 columns



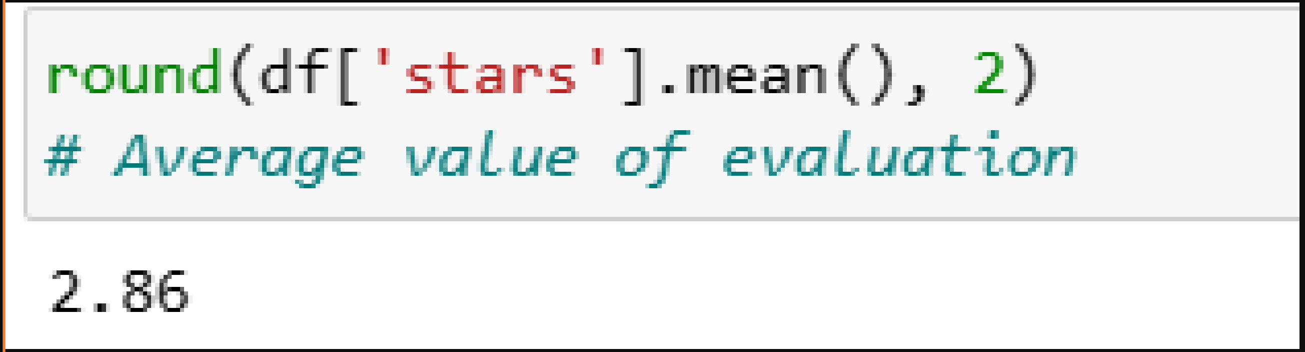
```
# Read the file CSV
df = pd.read_csv('Information_data Final Project.csv')

# Convert the column read_date in datatime
df['read_date'] = pd.to_datetime(df['read_date'])

# Filter only the rows for 2021
df_2021 = df[df['read_date'].dt.year == 2021]

# Now df_2021 contain only the rows of the year 2021
```

```
df.head(10)
```

	read_date	user_uuid	category	journalist_id	language	length	country	subscription_date	platform	article_id	stars
0	2021-02-25	243	art	117	it	short	it	24-8-2020	tablet	5128	3
1	2021-07-08	157	weather	111	it	long	it	12-2-2020	tablet	732766	5
2	2021-04-17	181	sport	114	en	short	uk	9-12-2020	pc	313130	1
3	2021-11-17	138	finance	111	it	short	it	6-4-2020	pc	612403	3
4	2021-10-04	94	news	103	it	long	it	24-3-2020	pc	632117	5
5	2021-07-27	69	finance	106	it	medium	it	23-8-2020	tablet	622487	4
6	2021-07-01	98	weather	103	it	long	it	26-9-2020	mobile	732585	5
7	2021-07-09	138	economy	108	fr	short	fr	6-4-2020	pc	211653	2
8	2021-05-13	206	economy	110	en	long	uk	7-11-2020	tablet	233755	2
9	2021-10-27	6	economy	118	en	long	uk	14-9-2020	tablet	233155	1

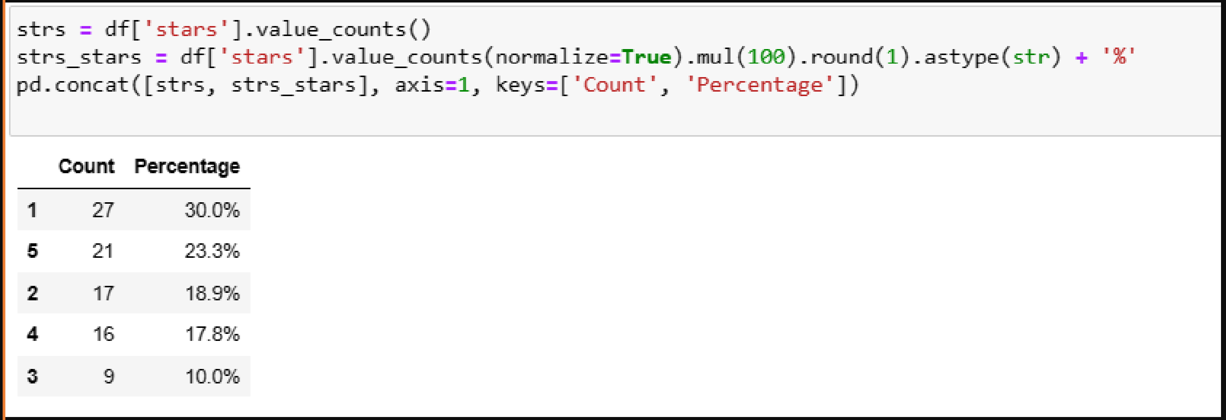
Python EDA

- Import the file and add the various libraries for analysis
- Extract the 2021's rows



```
round(df['stars'].mean(), 2)
# Average value of evaluation
```

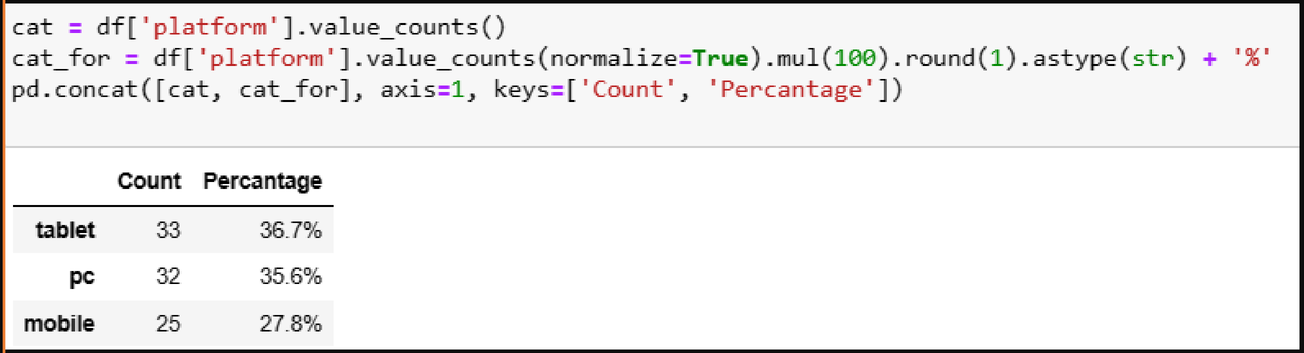
```
2.86
```



```
strs = df['stars'].value_counts()
strs_stars = df['stars'].value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.concat([strs, strs_stars], axis=1, keys=['Count', 'Percentage'])
```

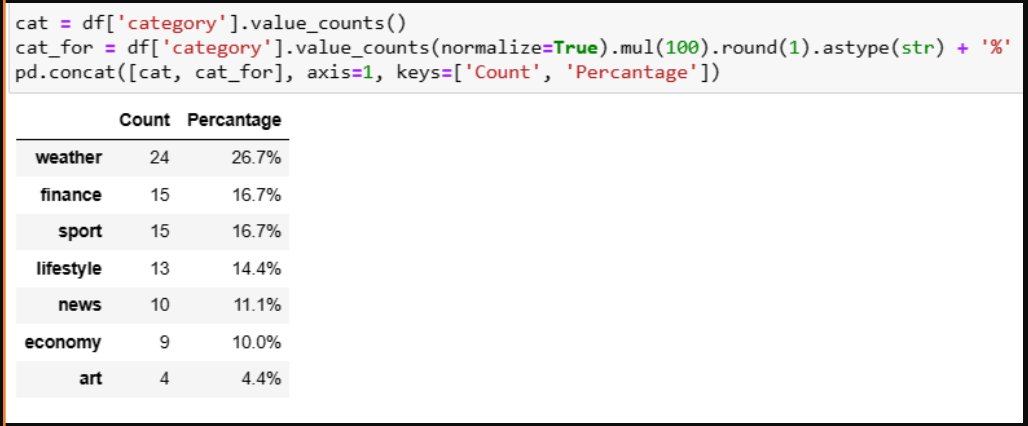
	Count	Percentage
1	27	30.0%
5	21	23.3%
2	17	18.9%
4	16	17.8%
3	9	10.0%

- We find average ratings for all the articles
- the code provides a count and percentage distribution of unique values in the 'stars' column of the DataFrame.



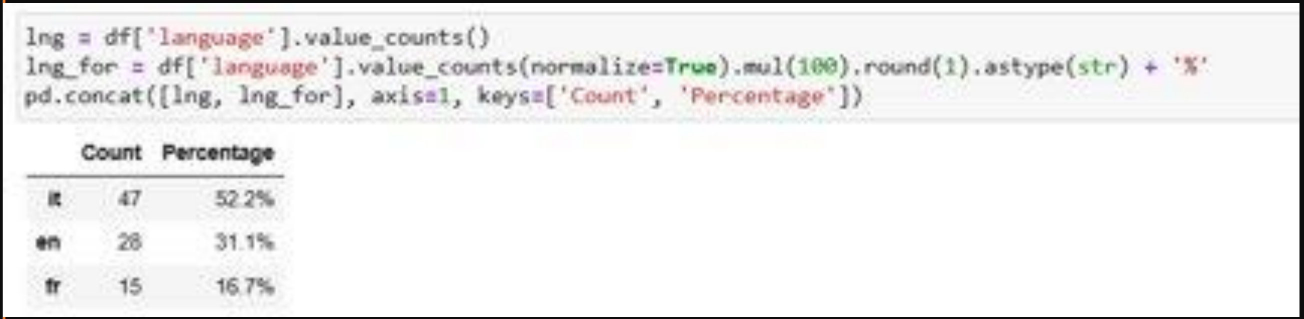
```python
cat = df['platform'].value_counts()
cat_for = df['platform'].value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.concat([cat, cat_for], axis=1, keys=['Count', 'Percantage'])
```

```
        Count  Percantage
tablet     33       36.7%
    pc     32       35.6%
mobile     25       27.8%
```



```python
cat = df['category'].value_counts()
cat_for = df['category'].value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.concat([cat, cat_for], axis=1, keys=['Count', 'Percantage'])
```

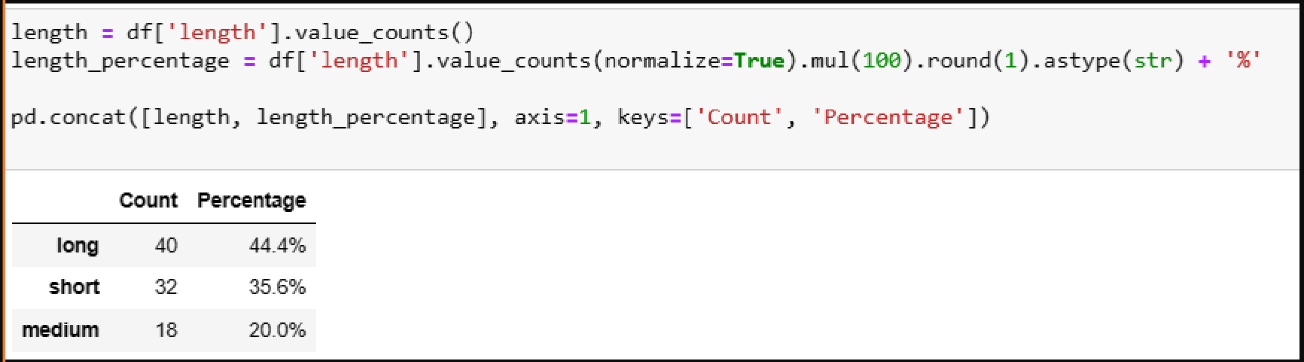
```
           Count  Percantage
  weather     24       26.7%
  finance     15       16.7%
    sport     15       16.7%
lifestyle     13       14.4%
     news     10       11.1%
  economy      9       10.0%
      art      4        4.4%
```



```python
lng = df['language'].value_counts()
lng_for = df['language'].value_counts(normalize=True).mul(100).round(1).astype(str) + '%'
pd.concat([lng, lng_for], axis=1, keys=['Count', 'Percentage'])
```

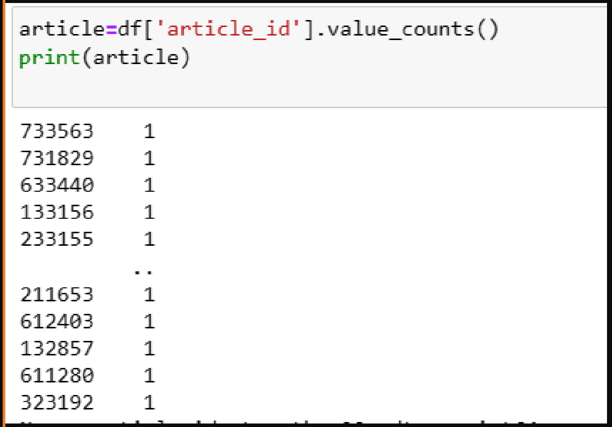
```
    Count  Percentage
it     47       52.2%
en     28       31.1%
fr     15       16.7%
```

- Let's continue with a series of Value Counts, to see the overall statistics for the various columns



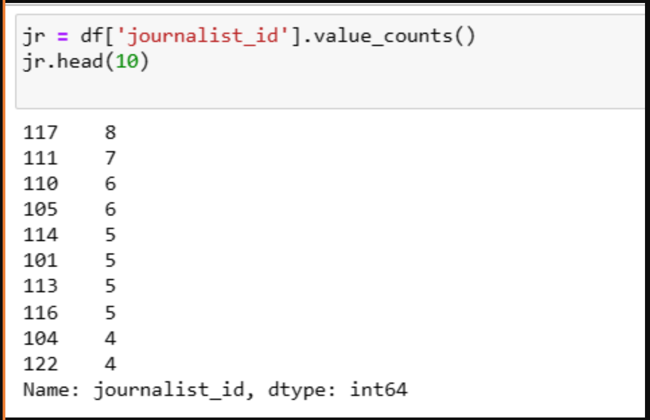
```python
length = df['length'].value_counts()
length_percentage = df['length'].value_counts(normalize=True).mul(100).round(1).astype(str) + '%'

pd.concat([length, length_percentage], axis=1, keys=['Count', 'Percentage'])
```

	Count	Percentage
long	40	44.4%
short	32	35.6%
medium	18	20.0%



```python
article=df['article_id'].value_counts()
print(article)
```

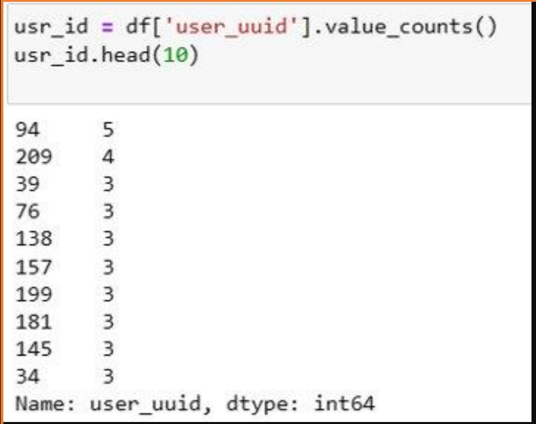
```
733563    1
731829    1
633440    1
133156    1
233155    1
         ..
211653    1
612403    1
132857    1
611280    1
323192    1
```



```python
jr = df['journalist_id'].value_counts()
jr.head(10)
```

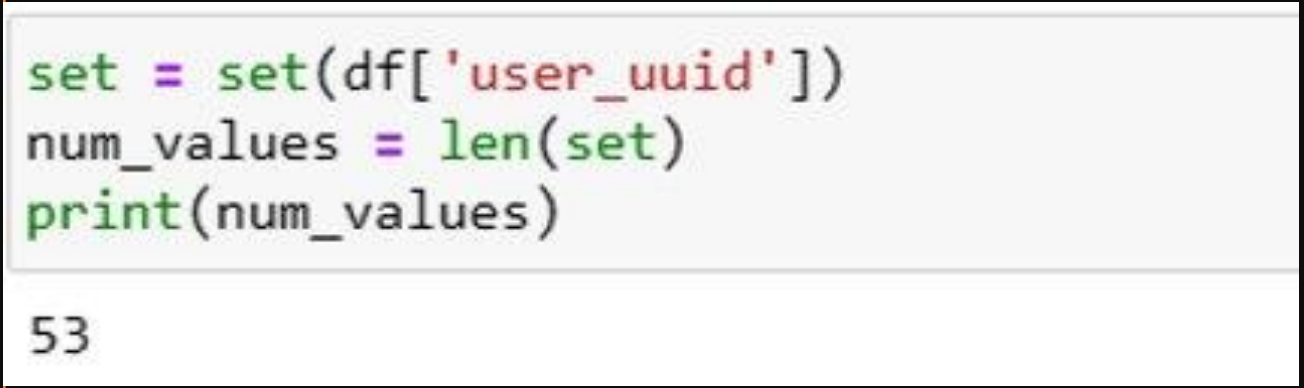
```
117    8
111    7
110    6
105    6
114    5
101    5
113    5
116    5
104    4
122    4
Name: journalist_id, dtype: int64
```

- We continue the check of the value counts also with the percentage of the articles lenght.



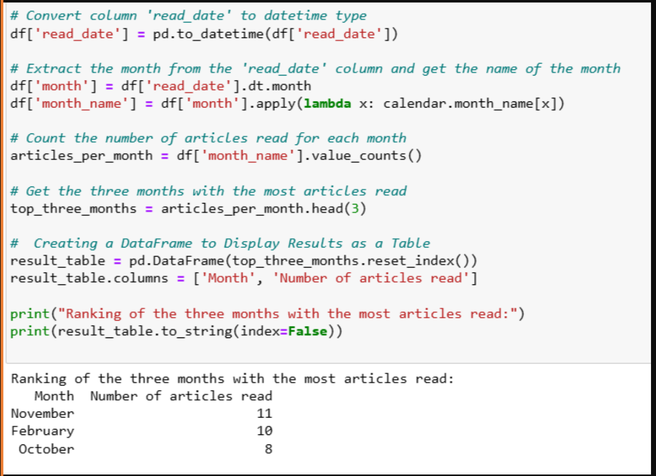
```
usr_id = df['user_uuid'].value_counts()
usr_id.head(10)
```

```
94     5
209    4
39     3
76     3
138    3
157    3
199    3
181    3
145    3
34     3
Name: user_uuid, dtype: int64
```



```
set = set(df['user_uuid'])
num_values = len(set)
print(num_values)
```

```
53
```

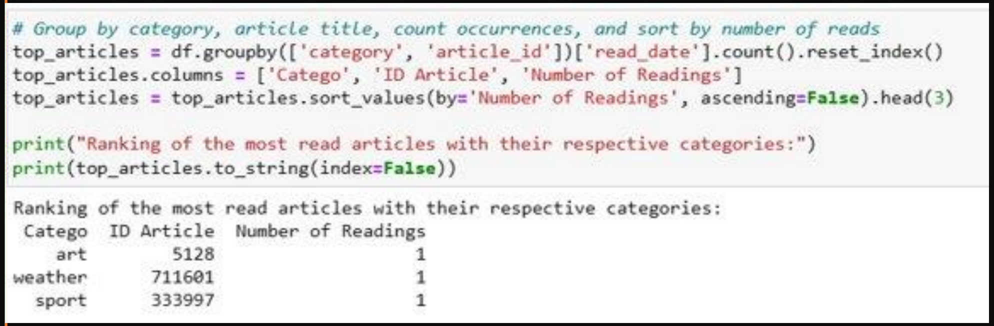


```
# Convert column 'read_date' to datetime type
df['read_date'] = pd.to_datetime(df['read_date'])

# Extract the month from the 'read_date' column and get the name of the month
df['month'] = df['read_date'].dt.month
df['month_name'] = df['month'].apply(lambda x: calendar.month_name[x])

# Count the number of articles read for each month
articles_per_month = df['month_name'].value_counts()

# Get the three months with the most articles read
top_three_months = articles_per_month.head(3)

#  Creating a DataFrame to Display Results as a Table
result_table = pd.DataFrame(top_three_months.reset_index())
result_table.columns = ['Month', 'Number of articles read']

print("Ranking of the three months with the most articles read:")
print(result_table.to_string(index=False))
```

```
Ranking of the three months with the most articles read:
   Month  Number of articles read
November                       11
February                       10
 October                        8
```

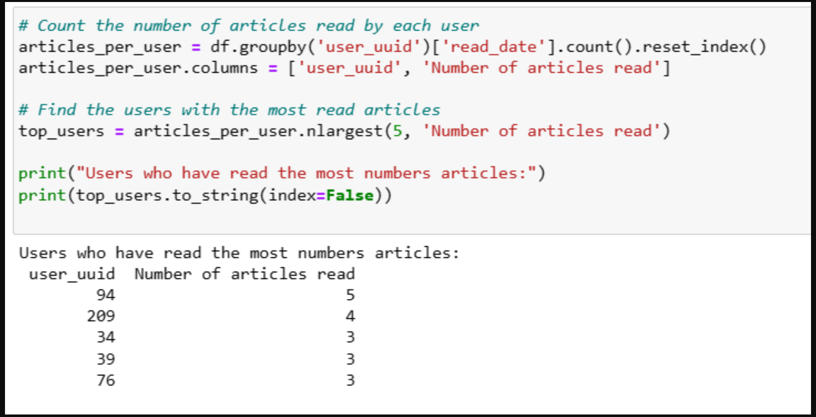
- Here we can see the number of annual readers

- And then we see the months which there were more readings



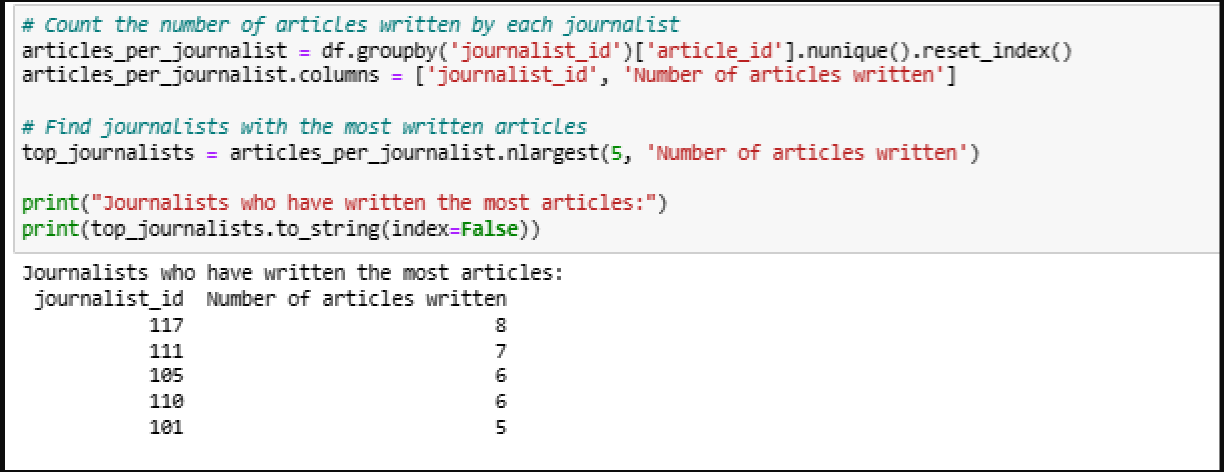
```
# Group by category, article title, count occurrences, and sort by number of reads
top_articles = df.groupby(['category', 'article_id'])['read_date'].count().reset_index()
top_articles.columns = ['Catego', 'ID Article', 'Number of Readings']
top_articles = top_articles.sort_values(by='Number of Readings', ascending=False).head(3)

print("Ranking of the most read articles with their respective categories:")
print(top_articles.to_string(index=False))
```

```
Ranking of the most read articles with their respective categories:
 Catego  ID Article  Number of Readings
    art        5128                   1
weather      711601                   1
  sport      333997                   1
```



```
# Count the number of articles read by each user
articles_per_user = df.groupby('user_uuid')['read_date'].count().reset_index()
articles_per_user.columns = ['user_uuid', 'Number of articles read']

# Find the users with the most read articles
top_users = articles_per_user.nlargest(5, 'Number of articles read')

print("Users who have read the most numbers articles:")
print(top_users.to_string(index=False))
```

```
Users who have read the most numbers articles:
 user_uuid  Number of articles read
        94                        5
       209                        4
        34                        3
        39                        3
        76                        3
```



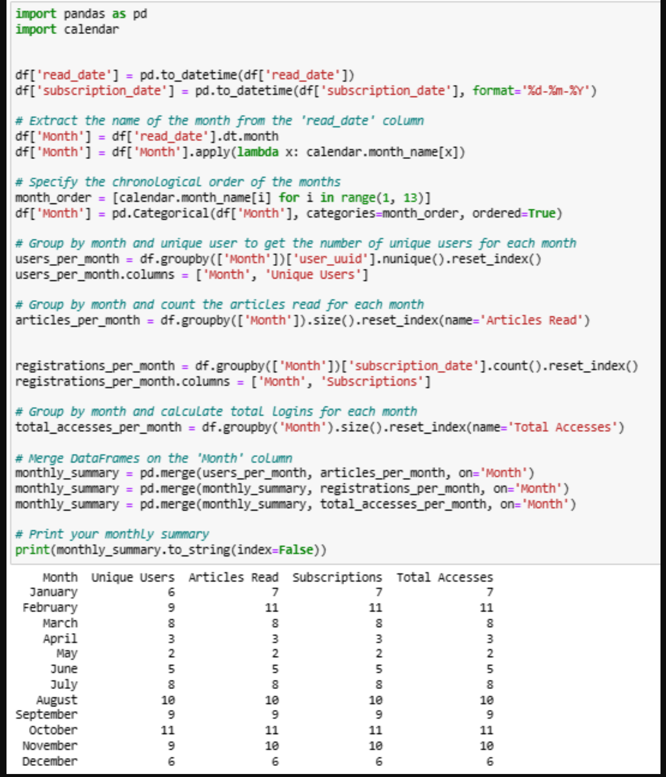
```
# Count the number of articles written by each journalist
articles_per_journalist = df.groupby('journalist_id')['article_id'].nunique().reset_index()
articles_per_journalist.columns = ['journalist_id', 'Number of articles written']

# Find journalists with the most written articles
top_journalists = articles_per_journalist.nlargest(5, 'Number of articles written')

print("Journalists who have written the most articles:")
print(top_journalists.to_string(index=False))
```

```
Journalists who have written the most articles:
 journalist_id  Number of articles written
           117                           8
           111                           7
           105                           6
           110                           6
           101                           5
```

- We keep going with the most read articles
- The users who reads the most numbers of articles
- And the journalist who have written the most number of articles

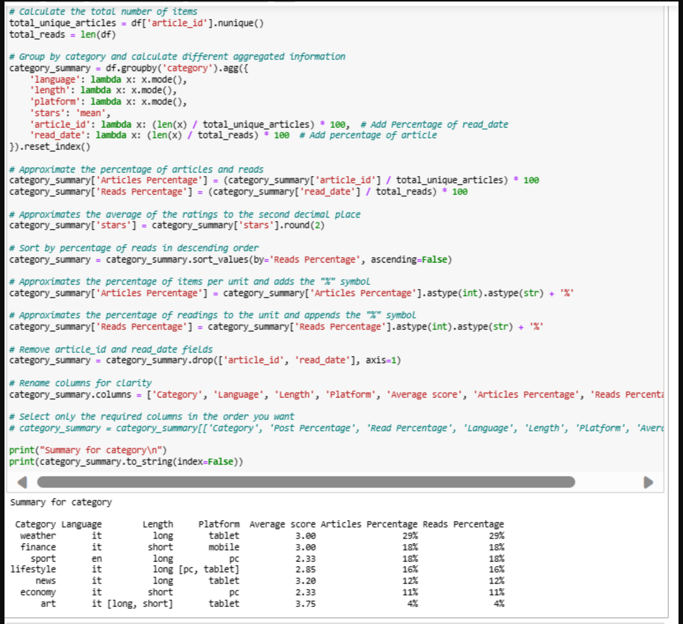
- We created a calender to see the unique users, articles read, subscriptions and total access for each month.

- The table below provides a summary of different categories, including information on language, article length, platform usage, average score, and percentage distribution of articles and reads within each category.



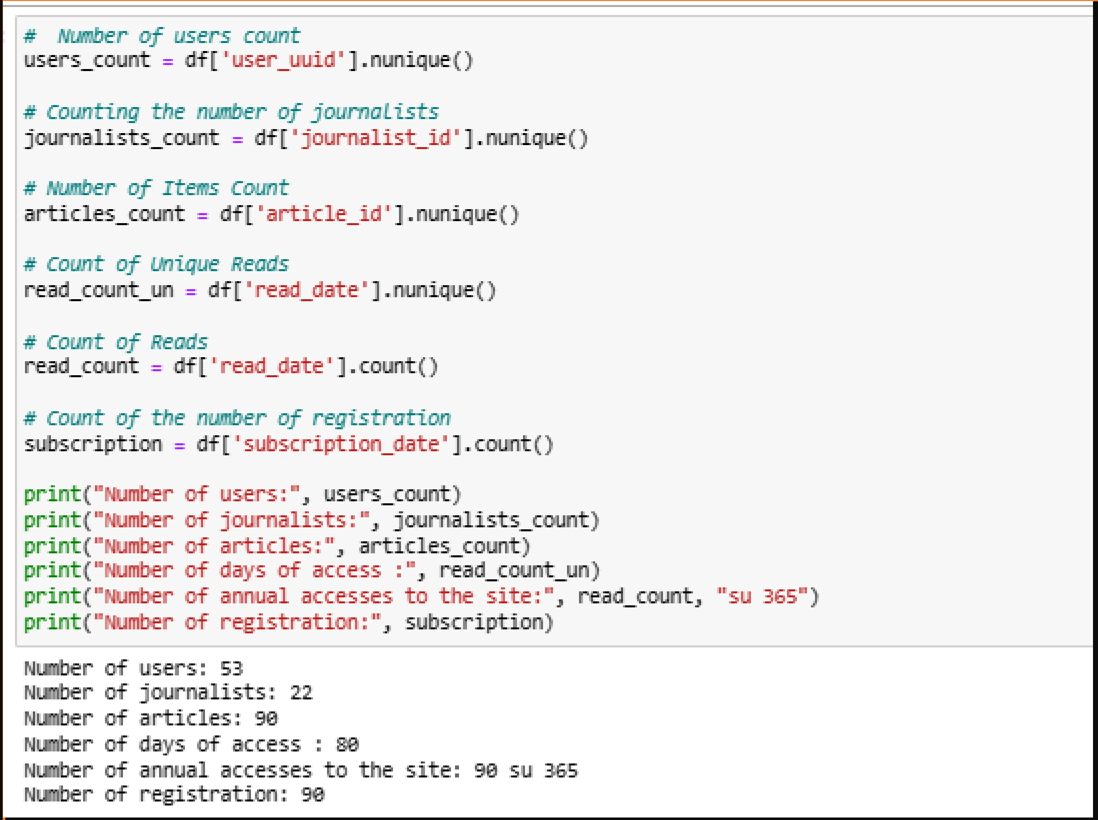
```python
import pandas as pd
import calendar


df['read_date'] = pd.to_datetime(df['read_date'])
df['subscription_date'] = pd.to_datetime(df['subscription_date'], format='%d-%m-%Y')

# Extract the name of the month from the 'read_date' column
df['Month'] = df['read_date'].dt.month
df['Month'] = df['Month'].apply(lambda x: calendar.month_name[x])

# Specify the chronological order of the months
month_order = [calendar.month_name[i] for i in range(1, 13)]
df['Month'] = pd.Categorical(df['Month'], categories=month_order, ordered=True)

# Group by month and unique user to get the number of unique users for each month
users_per_month = df.groupby(['Month'])['user_uuid'].nunique().reset_index()
users_per_month.columns = ['Month', 'Unique Users']

# Group by month and count the articles read for each month
articles_per_month = df.groupby(['Month']).size().reset_index(name='Articles Read')


registrations_per_month = df.groupby(['Month'])['subscription_date'].count().reset_index()
registrations_per_month.columns = ['Month', 'Subscriptions']

# Group by month and calculate total logins for each month
total_accesses_per_month = df.groupby('Month').size().reset_index(name='Total Accesses')

# Merge DataFrames on the 'Month' column
monthly_summary = pd.merge(users_per_month, articles_per_month, on='Month')
monthly_summary = pd.merge(monthly_summary, registrations_per_month, on='Month')
monthly_summary = pd.merge(monthly_summary, total_accesses_per_month, on='Month')

# Print your monthly summary
print(monthly_summary.to_string(index=False))
```

```
    Month  Unique Users  Articles Read  Subscriptions  Total Accesses
  January             6              7              7               7
 February             9             11             11              11
    March             8              8              8               8
    April             3              3              3               3
      May             2              2              2               2
     June             5              5              5               5
     July             8              8              8               8
   August            10             10             10              10
September             9              9              9               9
  October            11             11             11              11
 November             9             10             10              10
 December             6              6              6               6
```



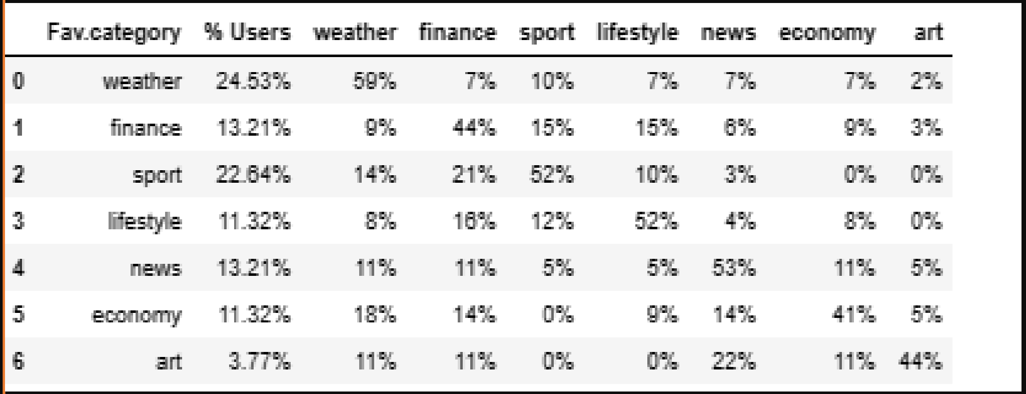
```python
# Calculate the total number of items
total_unique_articles = df['article_id'].nunique()
total_reads = len(df)

# Group by category and calculate different aggregated information
category_summary = df.groupby('category').agg({
    'language': lambda x: x.mode(),
    'length': lambda x: x.mode(),
    'platform': lambda x: x.mode(),
    'stars': 'mean',
    'article_id': lambda x: (len(x) / total_unique_articles) * 100,  # Add Percentage of read_date
    'read_date': lambda x: (len(x) / total_reads) * 100  # Add percentage of article
}).reset_index()

# Approximate the percentage of articles and reads
category_summary['Articles Percentage'] = (category_summary['article_id'] / total_unique_articles) * 100
category_summary['Reads Percentage'] = (category_summary['read_date'] / total_reads) * 100

# Approximates the average of the ratings to the second decimal place
category_summary['stars'] = category_summary['stars'].round(2)

# Sort by percentage of reads in descending order
category_summary = category_summary.sort_values(by='Reads Percentage', ascending=False)

# Approximates the percentage of items per unit and adds the "%" symbol
category_summary['Articles Percentage'] = category_summary['Articles Percentage'].astype(int).astype(str) + '%'

# Approximates the percentage of readings to the unit and appends the "%" symbol
category_summary['Reads Percentage'] = category_summary['Reads Percentage'].astype(int).astype(str) + '%'

# Remove article_id and read_date fields
category_summary = category_summary.drop(['article_id', 'read_date'], axis=1)

# Rename columns for clarity
category_summary.columns = ['Category', 'Language', 'Length', 'Platform', 'Average score', 'Articles Percentage', 'Reads Percenta

# Select only the required columns in the order you want
# category_summary = category_summary[['Category', 'Post Percentage', 'Read Percentage', 'Language', 'Length', 'Platform', 'Avera

print("Summary for category\n")
print(category_summary.to_string(index=False))
```

```
Summary for category

 Category Language         Length      Platform  Average score Articles Percentage Reads Percentage
  weather       it           long        tablet           3.00                 29%              29%
  finance       it          short        mobile           3.00                 18%              18%
    sport       en           long            pc           2.33                 18%              18%
lifestyle       it           long [pc, tablet]            2.85                 16%              16%
     news       it           long        tablet           3.20                 12%              12%
  economy       it          short            pc           2.33                 11%              11%
      art       it [long, short]        tablet            3.75                  4%               4%
```



```python
#  Number of users count
users_count = df['user_uuid'].nunique()

# Counting the number of journalists
journalists_count = df['journalist_id'].nunique()

# Number of Items Count
articles_count = df['article_id'].nunique()

# Count of Unique Reads
read_count_un = df['read_date'].nunique()

# Count of Reads
read_count = df['read_date'].count()

# Count of the number of registration
subscription = df['subscription_date'].count()

print("Number of users:", users_count)
print("Number of journalists:", journalists_count)
print("Number of articles:", articles_count)
print("Number of days of access :", read_count_un)
print("Number of annual accesses to the site:", read_count, "su 365")
print("Number of registration:", subscription)
```

```
Number of users: 53
Number of journalists: 22
Number of articles: 90
Number of days of access : 80
Number of annual accesses to the site: 90 su 365
Number of registration: 90
```



	Fav.category	% Users	weather	finance	sport	lifestyle	news	economy	art
0	weather	24.53%	59%	7%	10%	7%	7%	7%	2%
1	finance	13.21%	9%	44%	15%	15%	6%	9%	3%
2	sport	22.64%	14%	21%	52%	10%	3%	0%	0%
3	lifestyle	11.32%	8%	16%	12%	52%	4%	8%	0%
4	news	13.21%	11%	11%	5%	5%	53%	11%	5%
5	economy	11.32%	18%	14%	0%	9%	14%	41%	5%
6	art	3.77%	11%	11%	0%	0%	22%	11%	44%

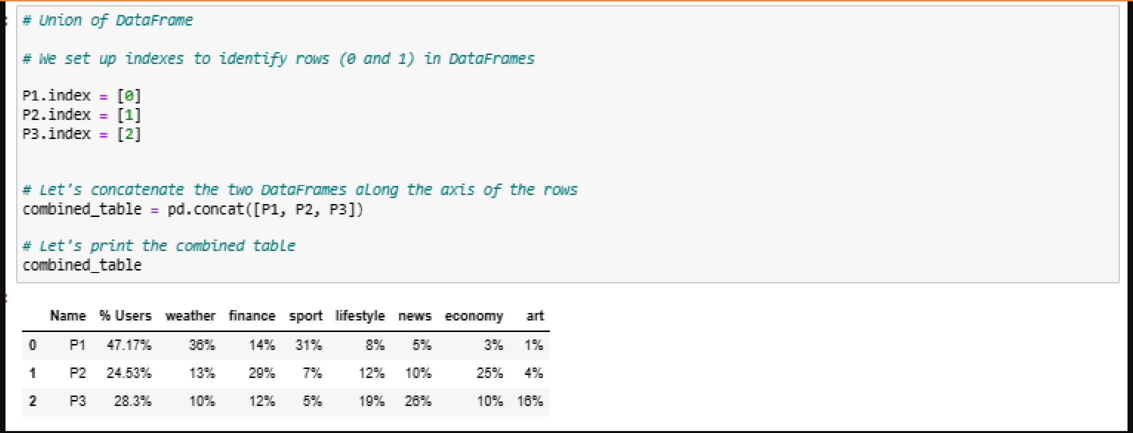
- The provided code performs a statistical analysis of a dataset, presenting key metrics related to user engagement and content on a platform.

- The second output provide a detailed overview of user engagement and content preferences, offering insights into the popularity of different categories and how users' reading interests are distributed within each main category.

- The provided code combines three DataFrames (P1, P2, and P3) into a single DataFrame named "combined_table." Each original DataFrame likely represents a summary or statistics for a specific persona (P1, P2, P3).

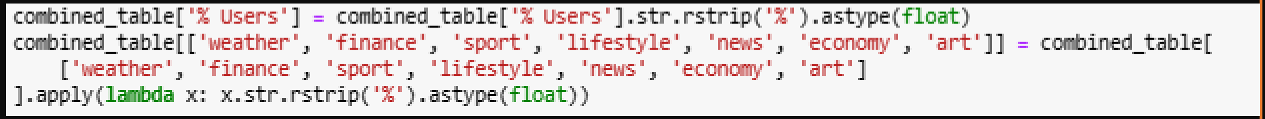
- The second image provided a code that modifies the "combined_table" DataFrame by converting the '%' values in the '% Users' column and specific category columns ('weather', 'finance', 'sport', 'lifestyle', 'news', 'economy', 'art') to numerical values.

- Then we save a new csv file

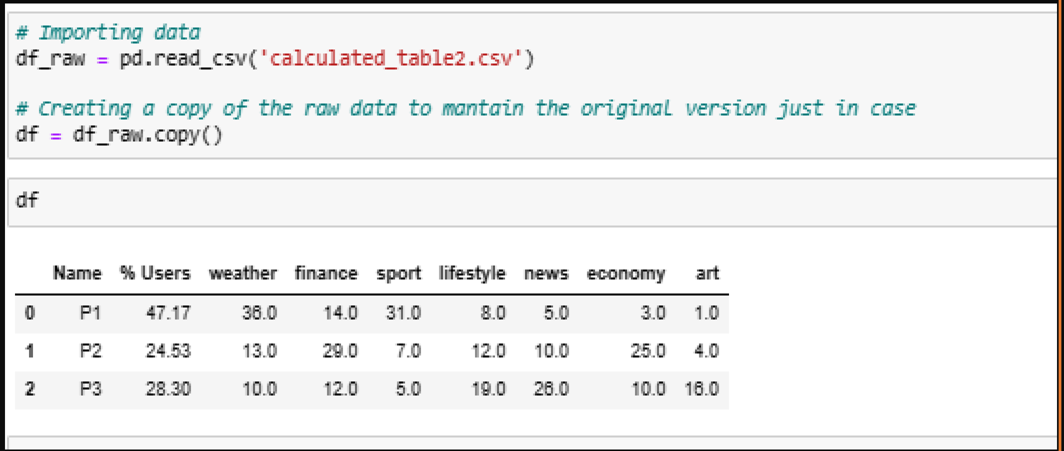


```
# Union of DataFrame

# We set up indexes to identify rows (0 and 1) in DataFrames

P1.index = [0]
P2.index = [1]
P3.index = [2]


# Let's concatenate the two DataFrames along the axis of the rows
combined_table = pd.concat([P1, P2, P3])

# Let's print the combined table
combined_table
```

	Name	% Users	weather	finance	sport	lifestyle	news	economy	art
0	P1	47.17%	36%	14%	31%	8%	5%	3%	1%
1	P2	24.53%	13%	29%	7%	12%	10%	25%	4%
2	P3	28.3%	10%	12%	5%	19%	26%	10%	16%



```
combined_table['% Users'] = combined_table['% Users'].str.rstrip('%').astype(float)
combined_table[['weather', 'finance', 'sport', 'lifestyle', 'news', 'economy', 'art']] = combined_table[
    ['weather', 'finance', 'sport', 'lifestyle', 'news', 'economy', 'art']
].apply(lambda x: x.str.rstrip('%').astype(float))
```

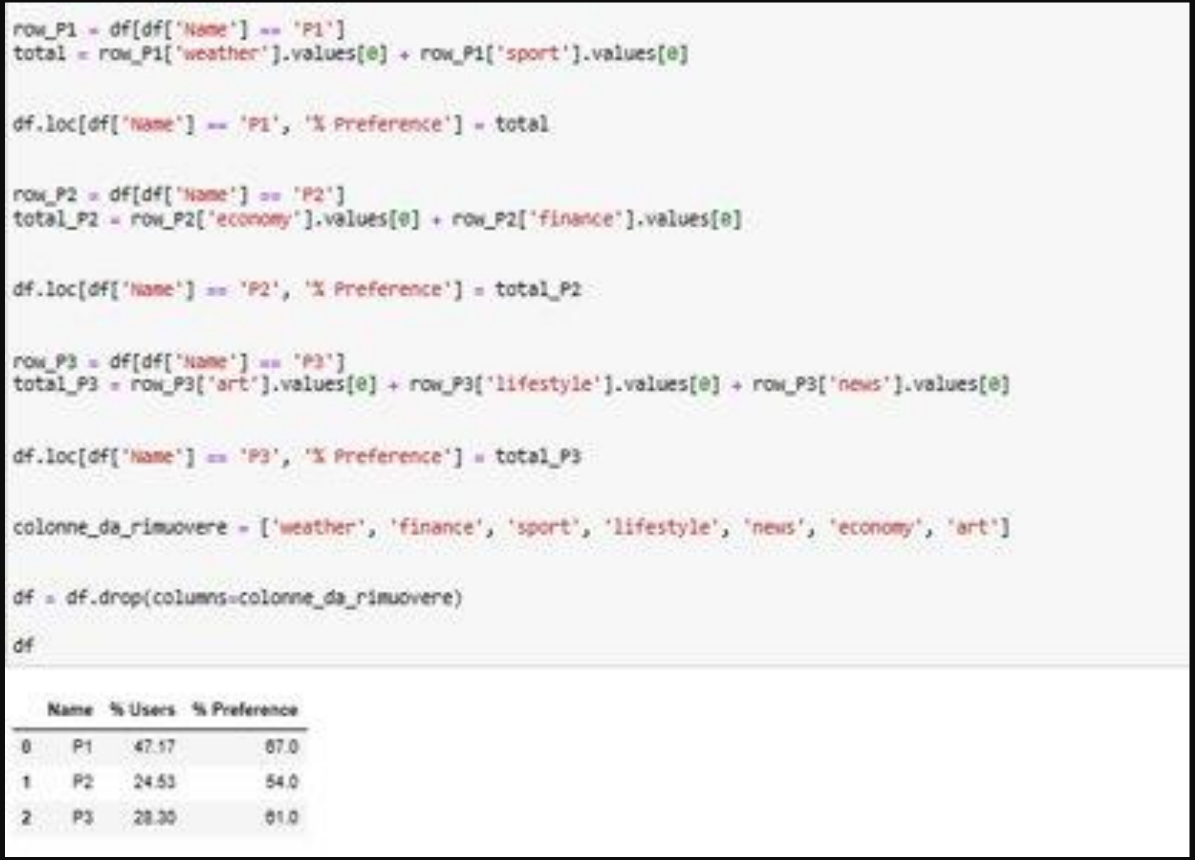


```
# Importing data
df_raw = pd.read_csv('calculated_table2.csv')

# Creating a copy of the raw data to mantain the original version just in case
df = df_raw.copy()
```

```
df
```

	Name	% Users	weather	finance	sport	lifestyle	news	economy	art
0	P1	47.17	36.0	14.0	31.0	8.0	5.0	3.0	1.0
1	P2	24.53	13.0	29.0	7.0	12.0	10.0	25.0	4.0
2	P3	28.30	10.0	12.0	5.0	19.0	26.0	10.0	16.0

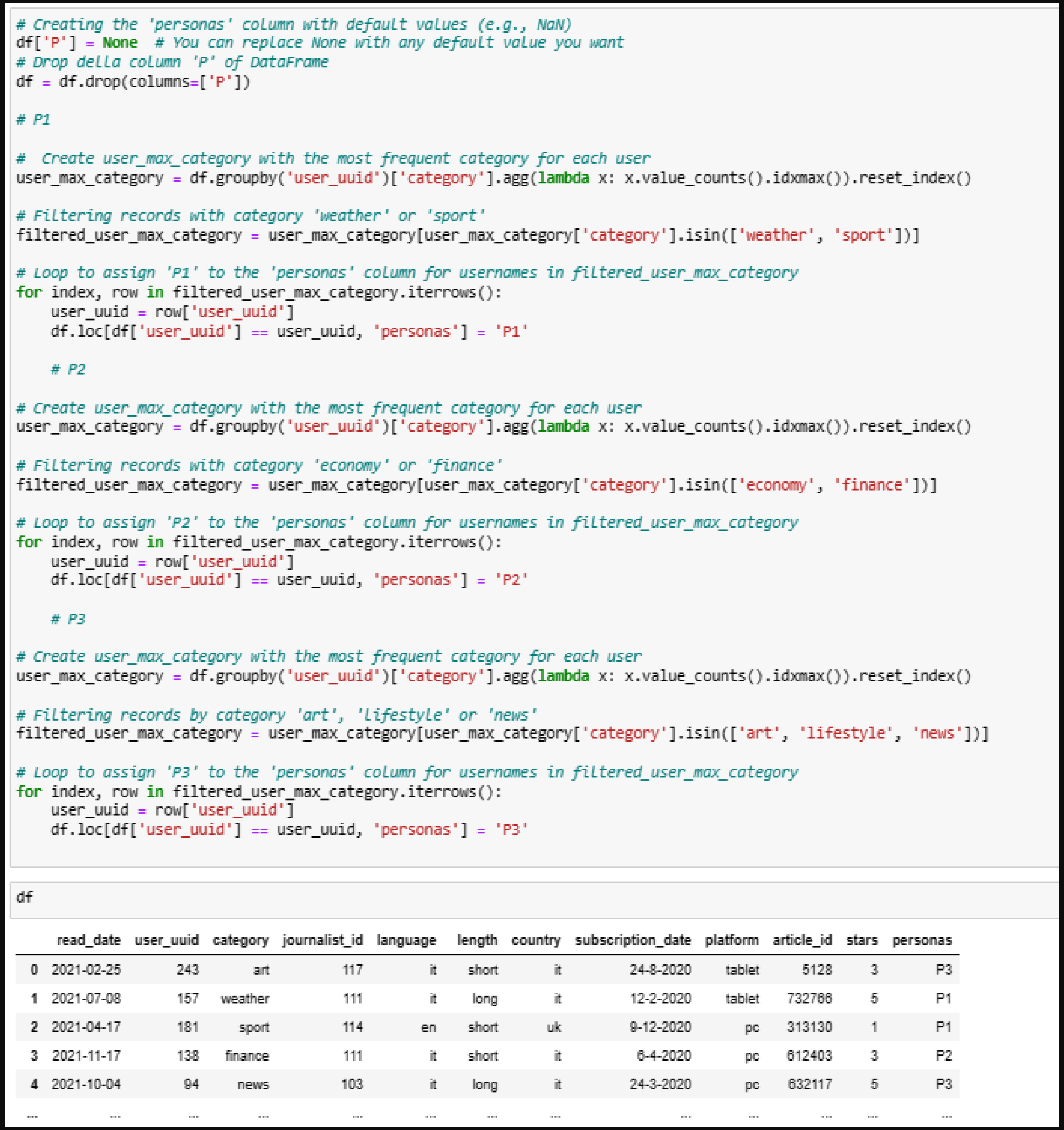
- The code calculates and assigns the total percentage preference for different categories to each persona (P1, P2, P3) based on specified columns in a DataFrame.



```
row_P1 = df[df['Name'] == 'P1']
total = row_P1['weather'].values[0] + row_P1['sport'].values[0]


df.loc[df['Name'] == 'P1', '% Preference'] = total


row_P2 = df[df['Name'] == 'P2']
total_P2 = row_P2['economy'].values[0] + row_P2['finance'].values[0]


df.loc[df['Name'] == 'P2', '% Preference'] = total_P2


row_P3 = df[df['Name'] == 'P3']
total_P3 = row_P3['art'].values[0] + row_P3['lifestyle'].values[0] + row_P3['news'].values[0]


df.loc[df['Name'] == 'P3', '% Preference'] = total_P3


colonne_da_rimuovere = ['weather', 'finance', 'sport', 'lifestyle', 'news', 'economy', 'art']


df = df.drop(columns=colonne_da_rimuovere)

df
```

```
   Name  % Users  % Preference
0    P1    47.17          67.0
1    P2    24.53          54.0
2    P3    28.30          61.0
```

- This code adds a new column 'personas' to a DataFrame with default values. It then determines the most frequent category for each user and assigns a persona ('P1', 'P2', or 'P3') based on specific category criteria.



```python
# Creating the 'personas' column with default values (e.g., NaN)
df['P'] = None  # You can replace None with any default value you want
# Drop della column 'P' of DataFrame
df = df.drop(columns=['P'])

# P1

#  Create user_max_category with the most frequent category for each user
user_max_category = df.groupby('user_uuid')['category'].agg(lambda x: x.value_counts().idxmax()).reset_index()

# Filtering records with category 'weather' or 'sport'
filtered_user_max_category = user_max_category[user_max_category['category'].isin(['weather', 'sport'])]

# Loop to assign 'P1' to the 'personas' column for usernames in filtered_user_max_category
for index, row in filtered_user_max_category.iterrows():
    user_uuid = row['user_uuid']
    df.loc[df['user_uuid'] == user_uuid, 'personas'] = 'P1'

    # P2

# Create user_max_category with the most frequent category for each user
user_max_category = df.groupby('user_uuid')['category'].agg(lambda x: x.value_counts().idxmax()).reset_index()

# Filtering records with category 'economy' or 'finance'
filtered_user_max_category = user_max_category[user_max_category['category'].isin(['economy', 'finance'])]

# Loop to assign 'P2' to the 'personas' column for usernames in filtered_user_max_category
for index, row in filtered_user_max_category.iterrows():
    user_uuid = row['user_uuid']
    df.loc[df['user_uuid'] == user_uuid, 'personas'] = 'P2'

    # P3

# Create user_max_category with the most frequent category for each user
user_max_category = df.groupby('user_uuid')['category'].agg(lambda x: x.value_counts().idxmax()).reset_index()

# Filtering records by category 'art', 'lifestyle' or 'news'
filtered_user_max_category = user_max_category[user_max_category['category'].isin(['art', 'lifestyle', 'news'])]

# Loop to assign 'P3' to the 'personas' column for usernames in filtered_user_max_category
for index, row in filtered_user_max_category.iterrows():
    user_uuid = row['user_uuid']
    df.loc[df['user_uuid'] == user_uuid, 'personas'] = 'P3'
```

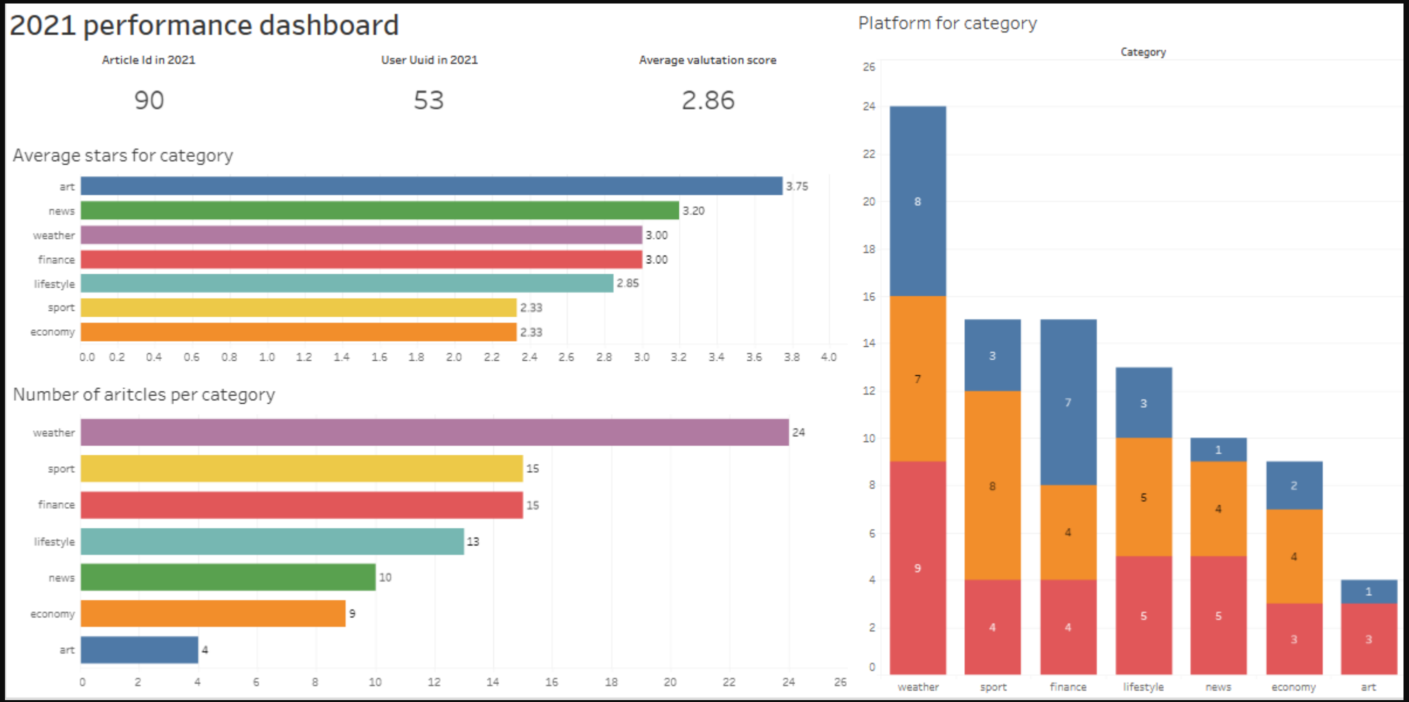
```python
df
```

	read_date	user_uuid	category	journalist_id	language	length	country	subscription_date	platform	article_id	stars	personas
0	2021-02-25	243	art	117	it	short	it	24-8-2020	tablet	5128	3	P3
1	2021-07-08	157	weather	111	it	long	it	12-2-2020	tablet	732766	5	P1
2	2021-04-17	181	sport	114	en	short	uk	9-12-2020	pc	313130	1	P1
3	2021-11-17	138	finance	111	it	short	it	6-4-2020	pc	612403	3	P2
4	2021-10-04	94	news	103	it	long	it	24-3-2020	pc	632117	5	P3
...	...	...	...	...	...	...	...	...	...	...	...	...

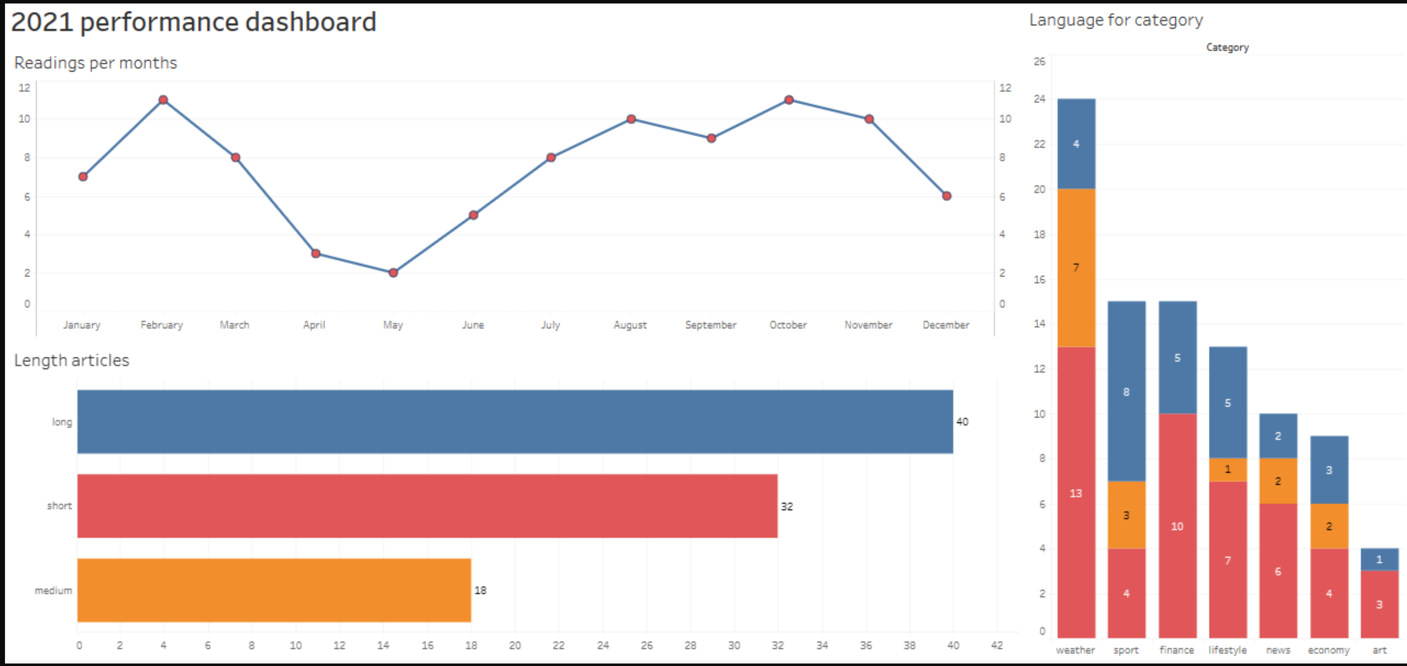
Tableau and creation of personas

At this point we created a series of dashboards on Tableau, first with a general performance of the site, then we going to make a dashboard for each personas.

We can start to see the first part of the general trend of the site, including the KPI's, the average stars for category, the number of articles per category and the platforms for category.

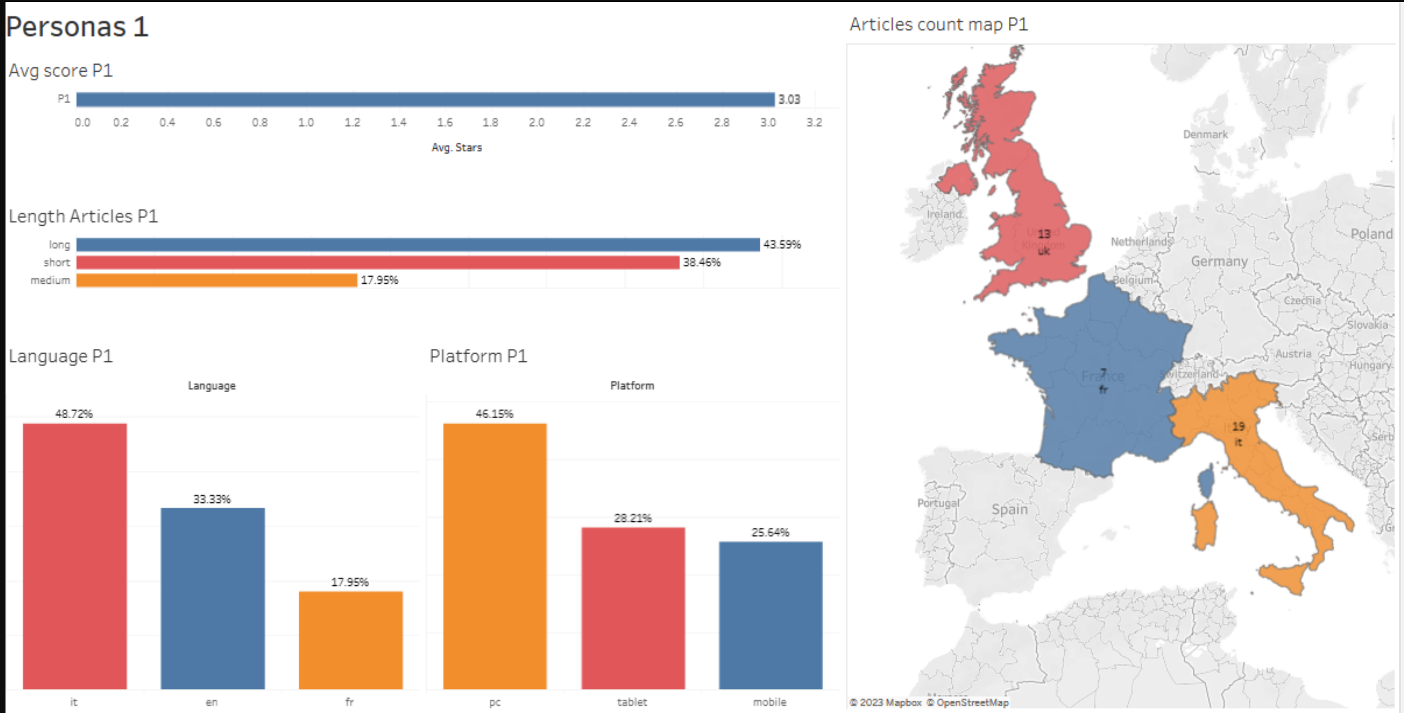


Let's continue with the readings per months charts, the lenght articles, and the language for category.

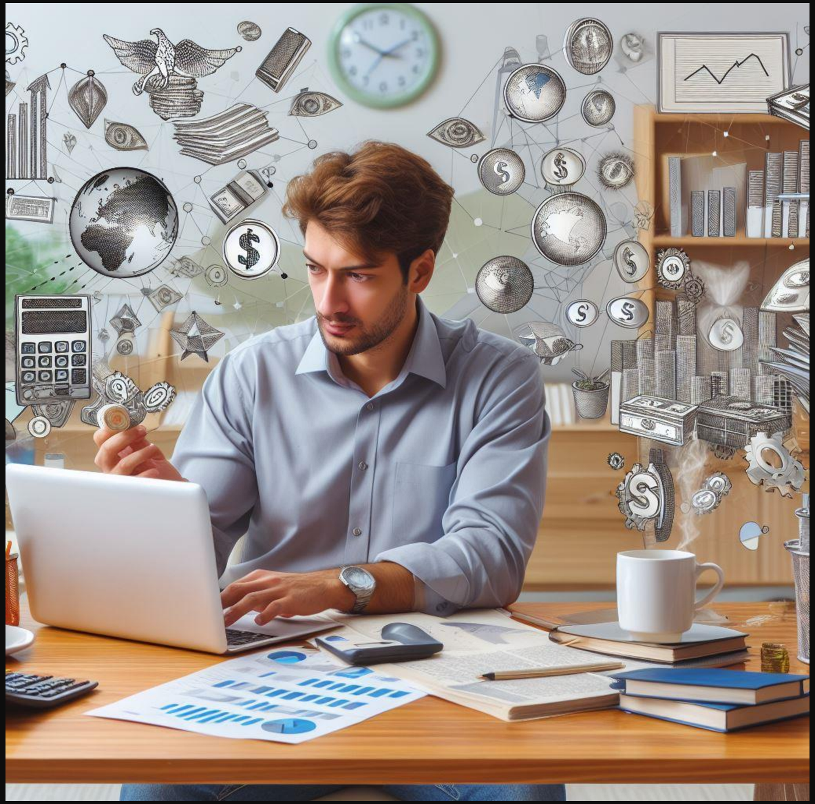


Personas 1

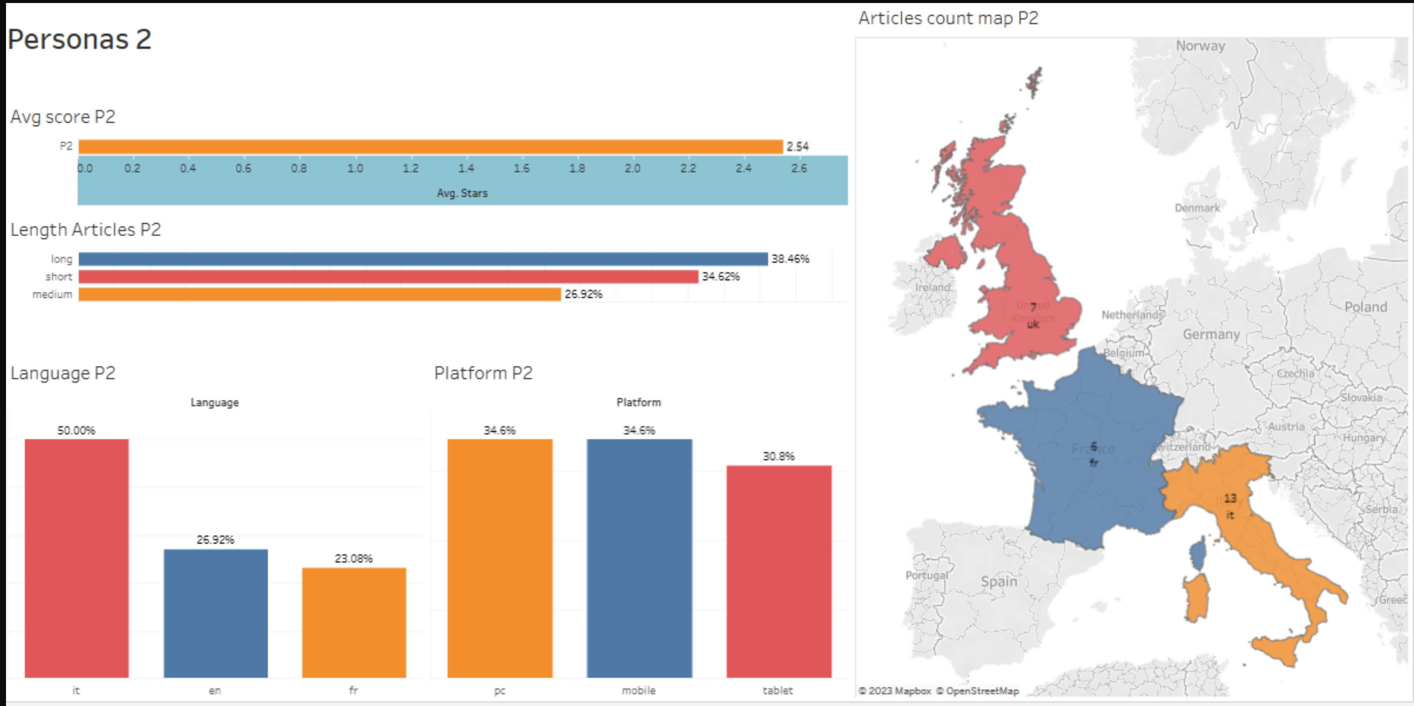
Kevin is the reader who reads sports and weather articles, his ratings are slightly above average, he prefers to read long articles but shorts as well, he reads mainly articles in Italian and loves to read these on the PC, most of the articles he reads are Italian, but also a good part of British.



Personas 2

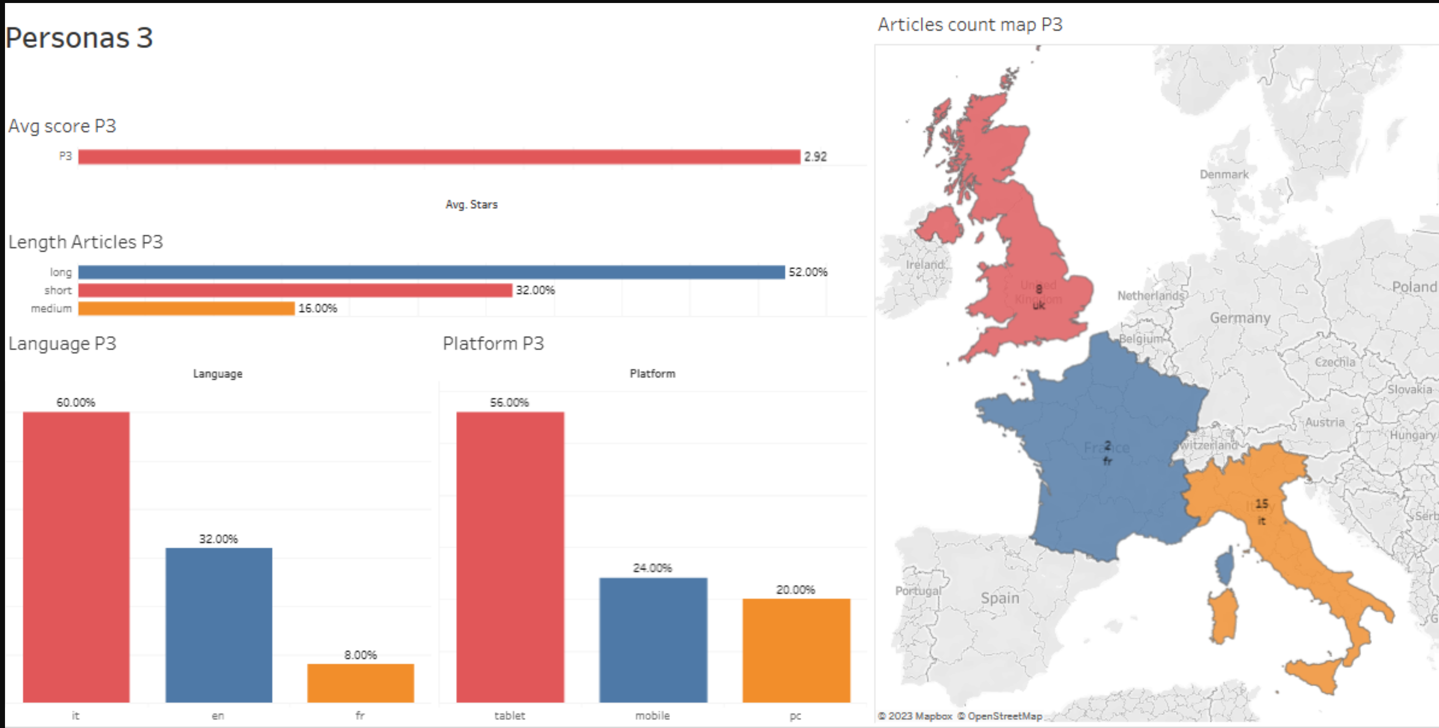


Christian is the user who reads economy and finance articles, his average valuation score is below 3 stars , he prefers to read both on pc or mobile, he likes the long articles but also shorts, he reads mainly articles in Italian.



Personas 3

Noemi is the user who reads news, lifestyle and art articles. She loves to read articles on tablet, she prefer to read in italian. She strongly prefers long articles, and her average valuation is slightly below 3 stars.



Strategy Personas 1

- Kevin is the type of user with the most potential. Its average ratings for articles turns out to be above 3 stars. It is the category that has the most articles in the countries. We could implement more articles in French, as well as targeting these on devices such as tablets and mobile to reach a greater number of readers. We could implement journalists who write medium-length articles.

Strategy Personas 2

- Christian has good potential as a category, it has a decent number of articles, but it could expand more for French and English readers. In addition, it's wise implementing the articles to an audience that reads predominantly on the tablet. It is advisable to re-evaluate the quality of the articles, given the low average rating, as well as implementing medium-length articles which are in the minority.

Strategy Personas 3

- Noemi is the type of user where it is necessary to intervene most urgently. We recommend an increase in the number of articles, especially in French, where there is an almost total absence with only 2 articles. As well as an increase in medium-length articles, which at the moment are the least. It would be good to advertise the articles for platforms such as mobile and tablet, to increase users on those as well. The average score from this type of user is also not the best, so it would be appropriate to improve the quality of the articles.

Site Strategy

- The first thing to do is increase the number of users by maybe 50%. For this reason, it would be excellent to increase the number of art articles, but also on economics, lifestyle and news that are in the minority. We recommend an improvement in the quality of items for the lifestyle, sports and economy categories, which have an average rating of less than 3 stars. The platform least used by users is mobile, you could create a convenient app for greater reading on the mobile device, and advertise it. The public tends to read constantly throughout the year, with peaks in February and October. Lows are reached in April and May, with 3 and 2 readers respectively. In these months it would be necessary to increase advertising reach, in order to generate more traffic. It's also wise to increase medium-sized articles, which are in the minority in almost all categories. Finally, it is appropriate to strongly increase the articles in French, which is completely absent in some categories.

THANK YOU

- jupyter notebook
- workbook tableau