Phase 4: Data Exploration

Team Report by Team OLAPPED CSCI 6401-01

Team Name :- OLAPPED

Course :- Data Mining

Instructor :- Prof. Shivanjali Khare

Course ID :- CSCI-6401-01

Session :- Fall 2023

Assignment :- Phase 4: Data Exploration

1. Team Name :- OLAPPED

GitHub Repo Link :- https://github.com/svarg1-unh/Fall-2023-Data-Mining

Team Member Names :-

(1) Sean Vargas – svarg1@unh.newhaven.edu

(2) Kaylie N Neal – kneal5@unh.newhaven.edu

(3) Prashant Rana – prana4@unh.newhaven.edu

(4) Rajdeep Bhattacharya – rbhat6@unh.newhaven.edu

2. Selected Dataset :-

Description of the selected dataset that we want to work with :-

The Data Set is named “trends” as it represents the Google Search Trends for a period of 20 years (2001 - 2020), pointing out the **[Top 5 Google Searches (Search Queries)]** by **[Categories]** with their **[Global Ranks]** and the ranks for the **[Top Countries]** by **[Year].**

It has 5 Attributes (represented by 5 columns) namely **location, year, category, rank and query**. And 26956 Data Points (represented by 26956 rows).

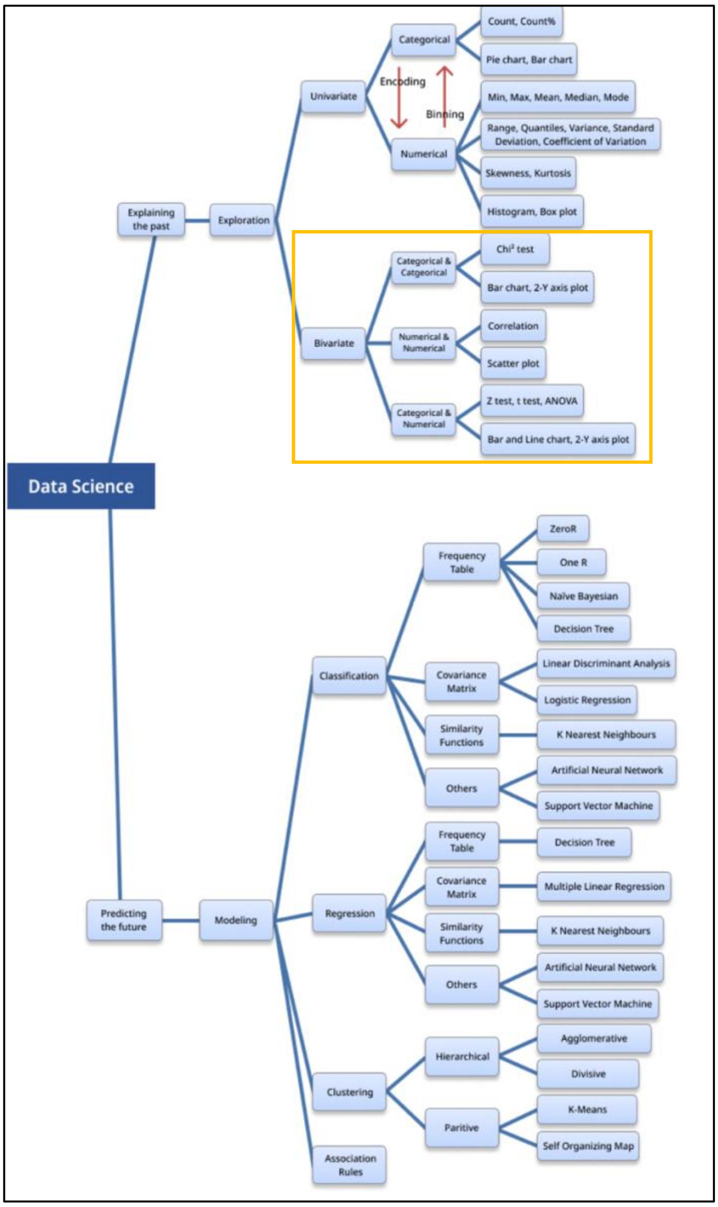
Research Question :-

The Research Question: - How can the patterns and trends in the **Google Search Engine Queries Dataset** be used to identify better rules for finding the most promising **Keywords (Search Queries)** and **Topics (Categories)** for showing more relevant **Search Results** in the future and targeting the relevant **SERPs (Search Engine Result Pages)** for **Ad Suggestions** by **timings (Seasonality)** and **locations (Geography)** for **Customer/Viewer Satisfaction** and higher **Ad Revenues**.

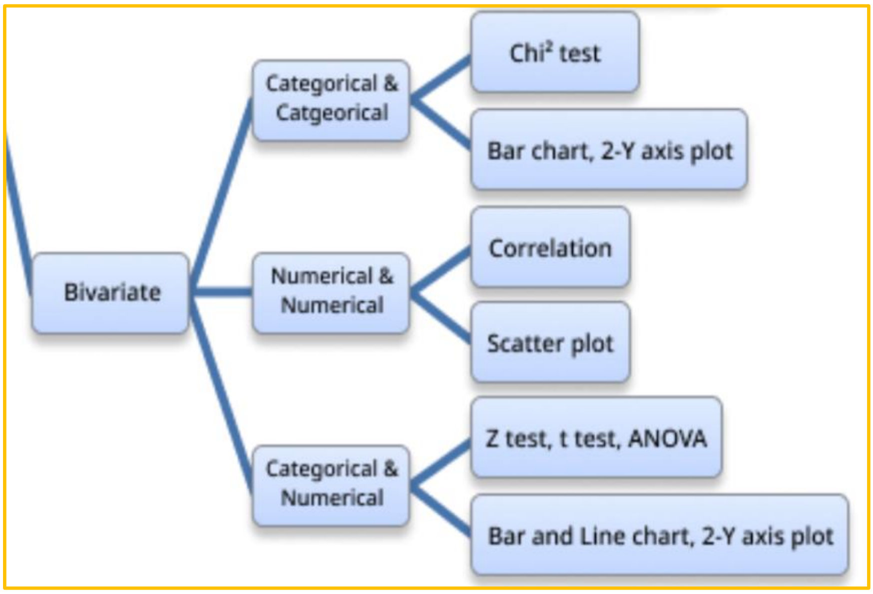
3. Our Exploration Journey :-

As was suggested in the Report suggestions we went over the flowchart of the exploration techniques listed in the link :-

https://www.saedsayad.com/data_mining_map.htm



As we are trying to establish a association between **Keywords (Search Queries)** and **Topics (Categories)** for showing more relevant **Search Results - Bivariate Analysis** was a natural selection. So, we focussed on the following section of the chart above: -



I. Bivariate Categorical - Categorical

We started off with Categorical - Categorical analysis, there were three categorical fields in our data namely Countries (Nominal), Ranks (Ordinal) and Search Words (Nominal).

Though we were searching for an association, but it mostly involved Time (Year) with these three fields. Finding association between these three won't do much help.

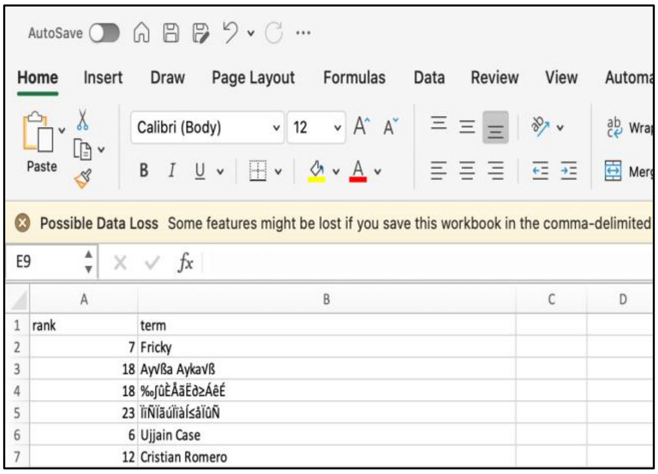
Also, as it was a huge data there were numerous data points scattered over a long X-Axis and Y-Axis.

For example: -

```
#Importing the necessary modules

import matplotlib.pyplot as plt

import pandas as pd



#Initializing the lists for X and Y

data = pd.read_csv('/Users/rajdeepbhattacharya/Desktop/Phase 4-Data Exploration-Assignment/trends-rank-
term.csv')
```

```
df = pd.DataFrame(data)

X = list(df.iloc[:, 1])
Y = list(df.iloc[:, 0])

#Plotting the data using bar() method
plt.bar(X, Y, color='g')
plt.title("Bar Graph On Terms And Ranks")
plt.xlabel("Terms")
plt.ylabel("Ranks")

#Showing the plot
plt.show()
```

Plotting Combination Bar Charts on the following: -

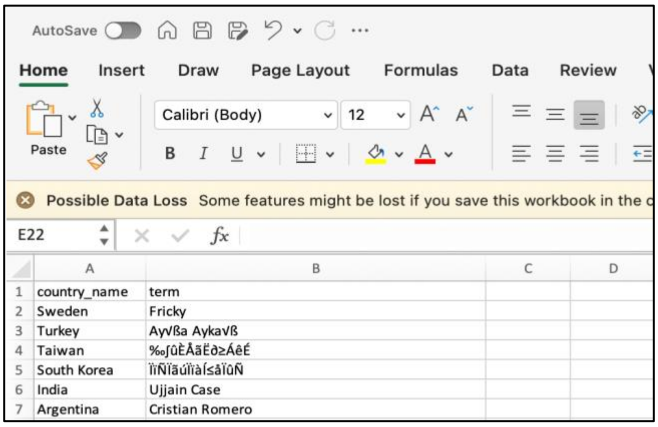


Possible Data Loss Some features might be lost if you save this workbook in the comma-delimited

E9

	A	B	C	D
1	rank	term		
2	7	Fricky		
3	18	Ay√ßa Ayka√ß		
4	18	‰∫ûÈÅãËð≥ÁêÉ		
5	23	ÏìÑÏãúÏïàÍ≤åÏûÑ		
6	6	Ujjain Case		
7	12	Cristian Romero		

And

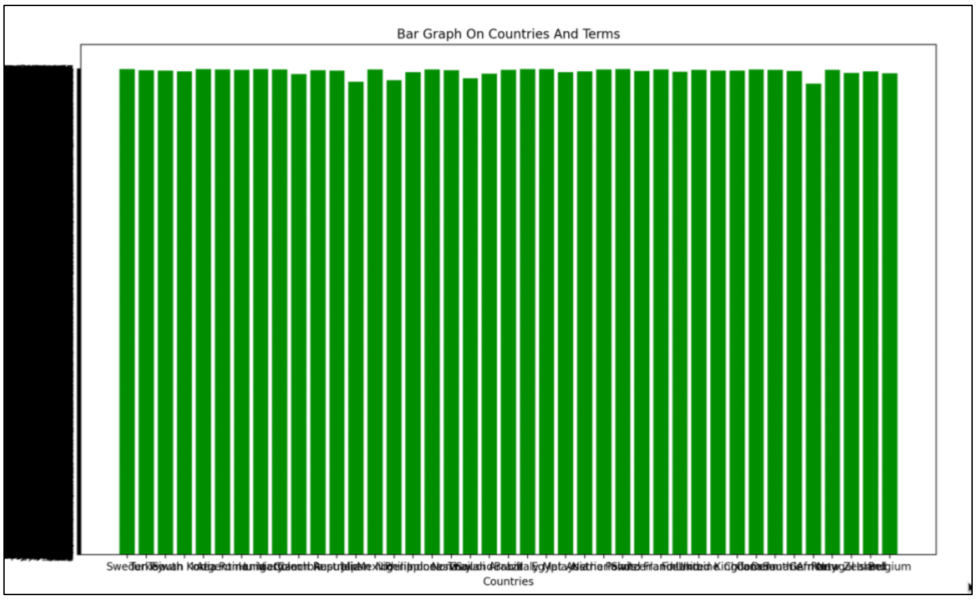


Possible Data Loss Some features might be lost if you save this workbook in the c

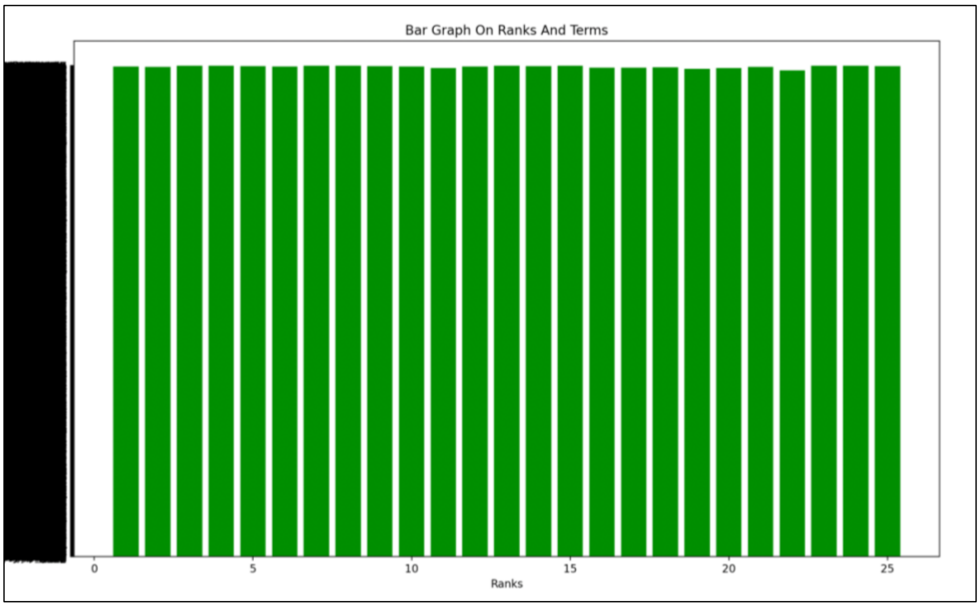
	A	B	C	D
1	country_name	term		
2	Sweden	Fricky		
3	Turkey	Ay√ßa Ayka√ß		
4	Taiwan	‰∫ûÈÅãË∂≥ÁêÉ		
5	South Korea	ÏïÑÏãúÏïàÍ≤åÏûÑ		
6	India	Ujjain Case		
7	Argentina	Cristian Romero		

Gave us: -

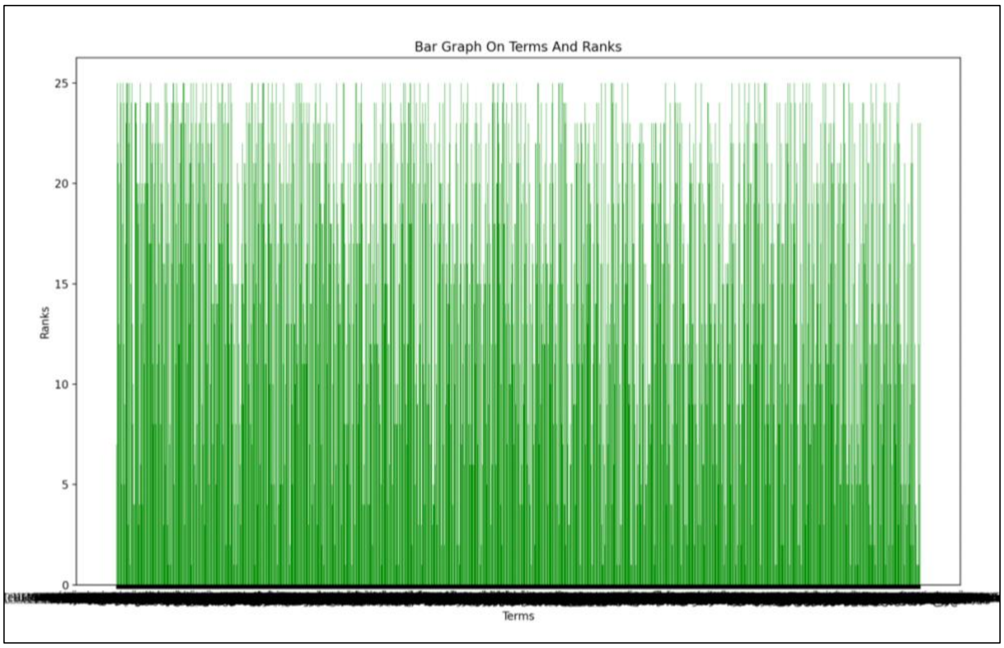
Search Terms (X-Axis) Vs Countries (Y-Axis).



Search Terms (X-Axis) Vs Ranks (Y-Axis).



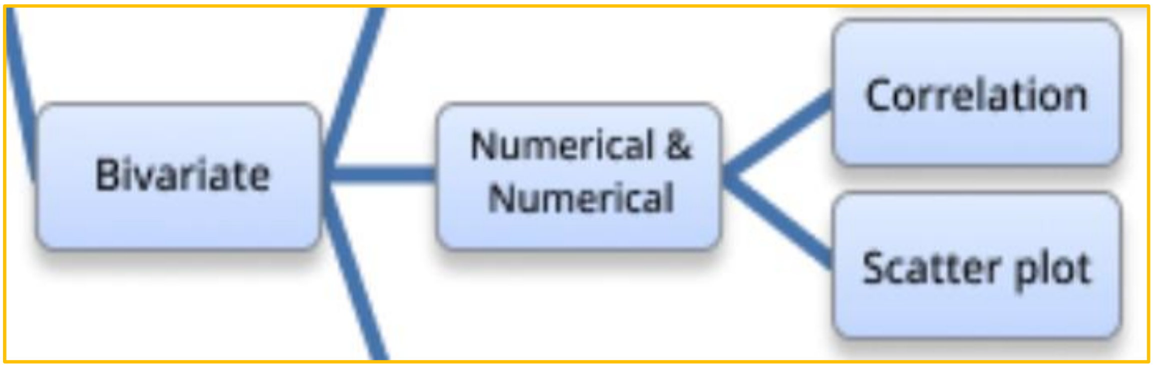
Ranks (X-Axis) Vs Search Terms (Y-Axis).



As from the plots you can see plotting Category Vs Category is pointless if we don't get bigger clublings. Also like I said Category Vs Category plots don't serve the objective (as derived from the research question).

II. Bivariate Numerical - Numerical

So, we moved on to the **Bivariate Numerical Vs Numerical plots**.



For Numerical - Numerical exploration, we used Scatterplot to plot Frequency of Search Terms against Different intervals of time. As Google tends to get tons of search queries, plotting all of them isn’t feasible. As our goal is to find the highest grossing search query, we first set an interval to get the highest frequency for each query by splitting time into intervals of month.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# %matplotlib inline
import warnings
warnings.filterwarnings('ignore')


df = pd.read_csv('./trends.csv')
df = df.drop(columns = ['region_name', 'region_code', 'score', 'refresh_date'], axis=1)
df['week'] = df['week'].str[:7]
df = df.rename(columns={'week':'year'})
```

As splitting time according to year would make our data sparse, we avoid that. Then we group our data according to Date interval and our search term to get the frequencies.

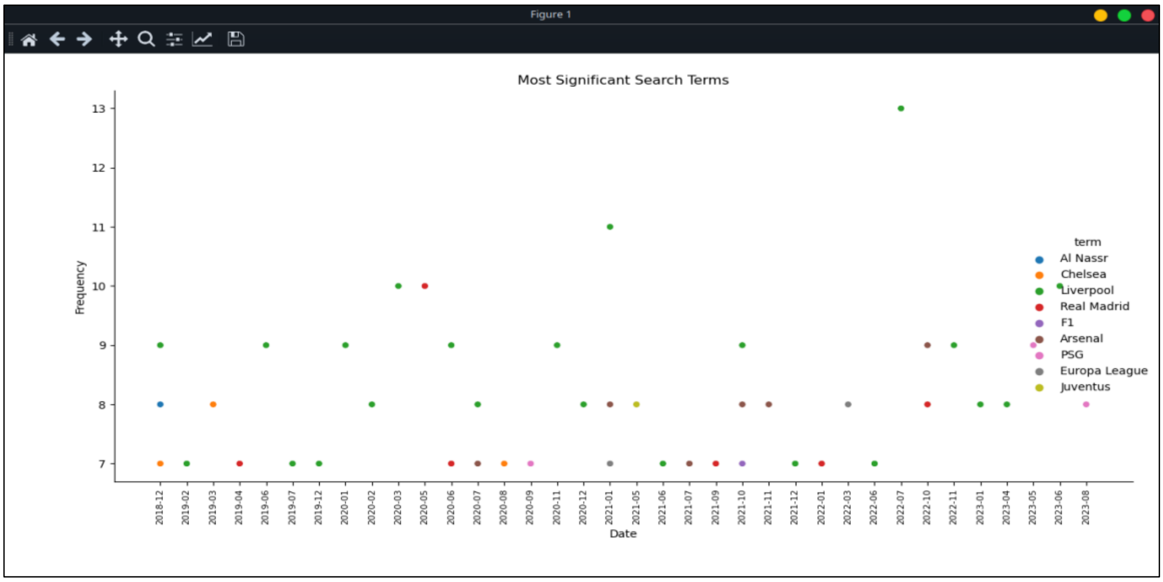
```
newDf = df.groupby(['year', 'term']).size().reset_index(name='Frequency')
```

As our target is to get data with high frequency, we filter our dataset with setting the limit of Frequency > 6.

```
newDf = newDf[newDf['Frequency'] > 6]
```

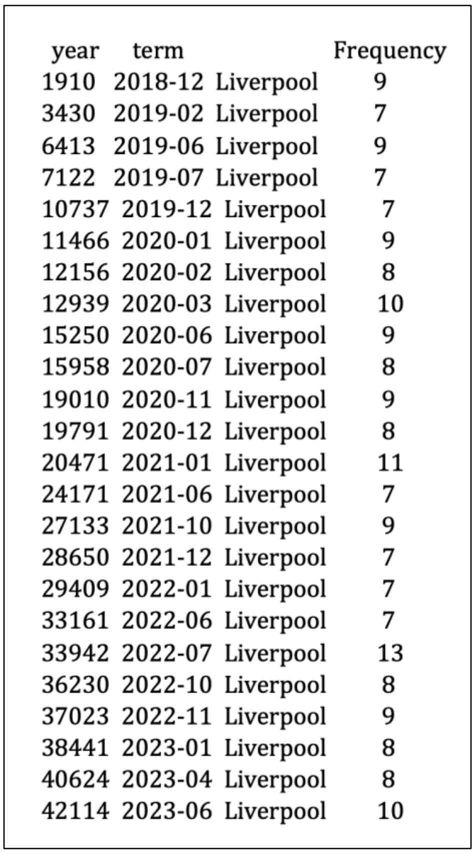
Now, we plot our scatterplot with Search frequency and date interval.

```
sns.relplot( data= newDf[['year','term', 'Frequency']], x = 'year', y = 'Frequency', hue = 'term')
plt.xlabel('Date')
plt.ylabel('Frequency')
plt.title('Most Significant Search Terms')
plt.xticks(rotation='vertical', fontsize=8)
plt.tight_layout()
plt.show()
```



For determining correlation between search term frequency and time, we isolate our dataset with data concerning only one search term suppose ‘Liverpool’.

```
liverpoolDf = newDf[newDf['term'] == 'Liverpool']

print(liverpoolDf)
```



```
  year    term              Frequency
 1910  2018-12  Liverpool      9
 3430  2019-02  Liverpool      7
 6413  2019-06  Liverpool      9
 7122  2019-07  Liverpool      7
 10737  2019-12  Liverpool      7
 11466  2020-01  Liverpool      9
 12156  2020-02  Liverpool      8
 12939  2020-03  Liverpool     10
 15250  2020-06  Liverpool      9
 15958  2020-07  Liverpool      8
 19010  2020-11  Liverpool      9
 19791  2020-12  Liverpool      8
 20471  2021-01  Liverpool     11
 24171  2021-06  Liverpool      7
 27133  2021-10  Liverpool      9
 28650  2021-12  Liverpool      7
 29409  2022-01  Liverpool      7
 33161  2022-06  Liverpool      7
 33942  2022-07  Liverpool     13
 36230  2022-10  Liverpool      8
 37023  2022-11  Liverpool      9
 38441  2023-01  Liverpool      8
 40624  2023-04  Liverpool      8
 42114  2023-06  Liverpool     10
```

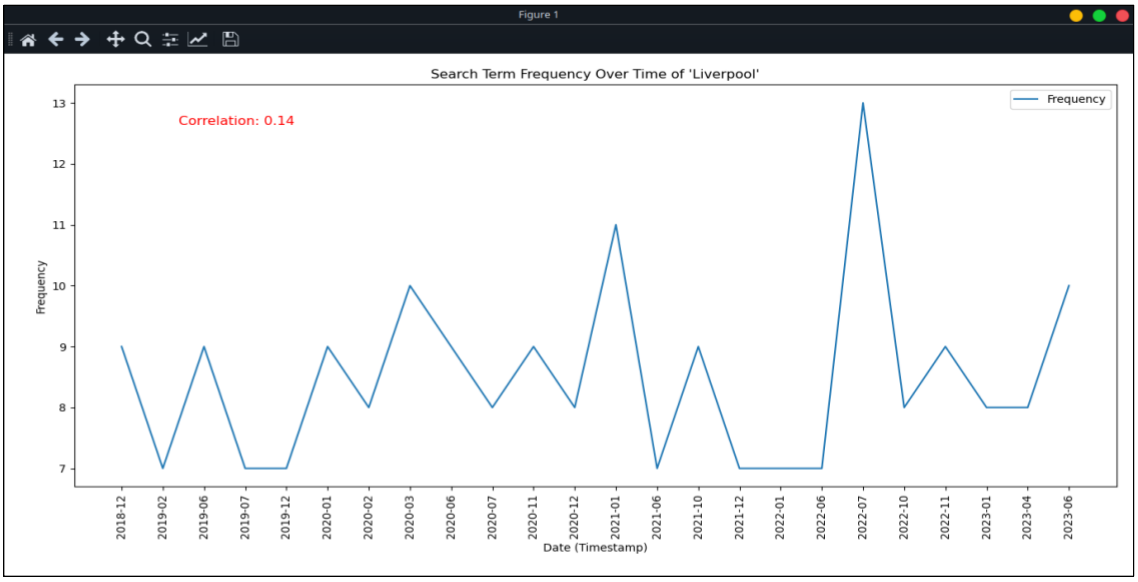
Then we convert our date interval into numeric value i.e. timestamp.

```
liverpoolDf['Numeric_date'] = pd.to_datetime(liverpoolDf['year']).apply(lambda x: x.timestamp())
```

After that we find our correlation and plot a line graph of search term frequency over interval of time.

```
correlation = liverpoolDf['Numeric_date'].corr(liverpoolDf['Frequency'])
print(f"Correlation: {correlation: .2f} ")
```

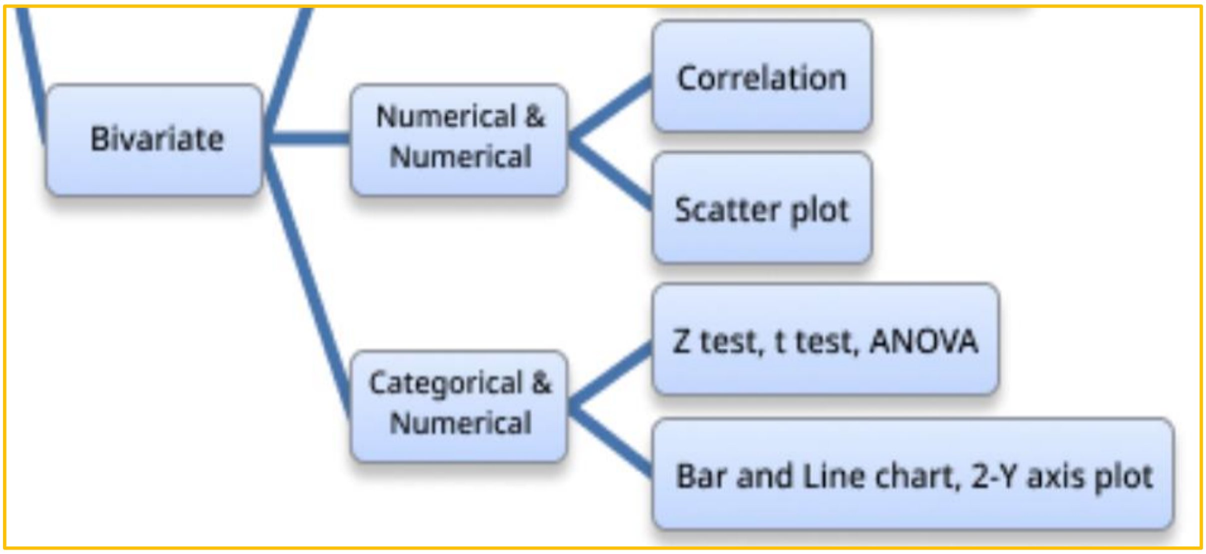
```
plt.figure(figsize=(10, 6))
plt.plot(liverpoolDf['year'], liverpoolDf['Frequency'], label='Frequency')
plt.xlabel('Date (Timestamp)')

plt.ylabel('Frequency')

plt.title('Search Term Frequency Over Time of \'Liverpool\'')



plt.legend()



# Display the plot


plt.text(0.1, 0.9, f'Correlation: {correlation:.2f}', transform=plt.gca().transAxes, fontsize=12, color='red')

plt.xticks(rotation='vertical')

plt.tight_layout()

plt.show()
```



Since, the correlation between search term frequency and date interval is 0.14, it seems less likely to be related.

III. Bivariate Numerical Vs Categorical

To explore further we delved into Bivariate Numerical Vs Categorical plots.



For numerical categorical exploration, we used a bar plot to plot the number of searches against the region of Europe where the search occurred. As Google gets search results from all over the world, trying to plot every country would be very time consuming and result in a hard-to-understand visualization. Since we are focusing on Europe for this exploration, we first need to filter our dataset for only search terms that came from European countries.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
# Data preprocessing
trends = pd.read_csv("../trends.csv")
trends = trends.drop(columns = ["country_code", "region_code", "refresh_date"], axis=1)
trends["week"] = trends["week"].str[:7]
trends = trends.rename(columns = {"week":"Year-Month"})
trends.columns = ["Score", "Year-Month", "Rank", "Country", "Sub-Region", "Term"]
trends = trends[["Term", "Rank", "Country", "Sub-Region", "Year-Month", "Score"]]
```

Once preprocessing is finished, we can filter the dataset for European countries.

```
countries = ["Sweden", "Turkey", "Romania", "Czech Republic", "Norway", \    "Italy",
"Austria", "Netherlands", "Poland", "Switzerland", \
"France", "Finland", "Ukraine", "United Kingdom", "Denmark", "Germany", \ "Portugal",
"Belgium"]
```

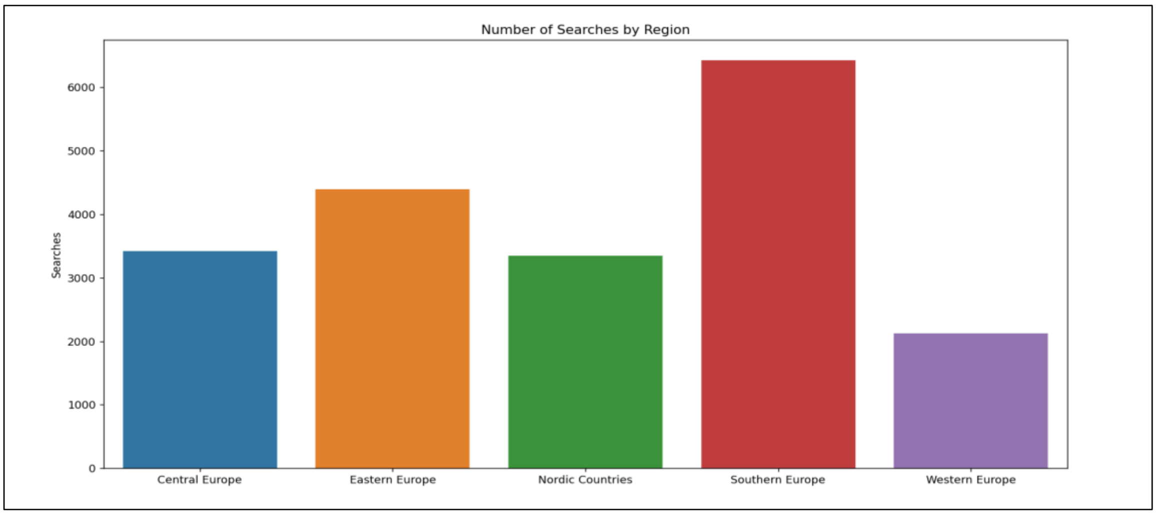
```
europeTrends = trends[trends["Country"].isin(countries)]
```

After we filter for European countries, we can then assign region classifiers to each entry.

```
nordicCountries = ["Sweden", "Norway", "Denmark", "Finland"]
eastEuroCountries = ["Romania", "Poland", "Ukraine"]
southEuroCountries = ["Turkey", "Italy", "Portugal"]
centralEuroCountries = ["Czech Republic", "Germany", "Austria", "Switzerland"]
westEuroCountries = ["Netherlands", "France", "United Kingdom", "Belgium"]
# Assign region classifiers to countries
europeTrends = europeTrends.assign(Region = europeTrends["Country"])
europeTrends.loc[europeTrends["Region"].isin(nordicCountries), "Region"] = "Nordic Countries"
europeTrends.loc[europeTrends["Region"].isin(eastEuroCountries), "Region"] = "Eastern Europe"
europeTrends.loc[europeTrends["Region"].isin(southEuroCountries), "Region"] = "Southern Europe"
europeTrends.loc[europeTrends["Region"].isin(centralEuroCountries), "Region"] = "Central Europe"
europeTrends.loc[europeTrends["Region"].isin(westEuroCountries), "Region"] = "Western Europe"
```

Now we can make our bar plot, using the count() function to aggregate searches by region.

```
# Plot number of searches by region
europeTrendsCount = europeTrends.groupby("Region", as_index=False).count()
sns.barplot(x="Region", y="Term", data=europeTrendsCount[["Region", "Term"]])
plt.xlabel("")
plt.ylabel("Searches")
plt.title("Number of Searches by Region")
plt.show()
```

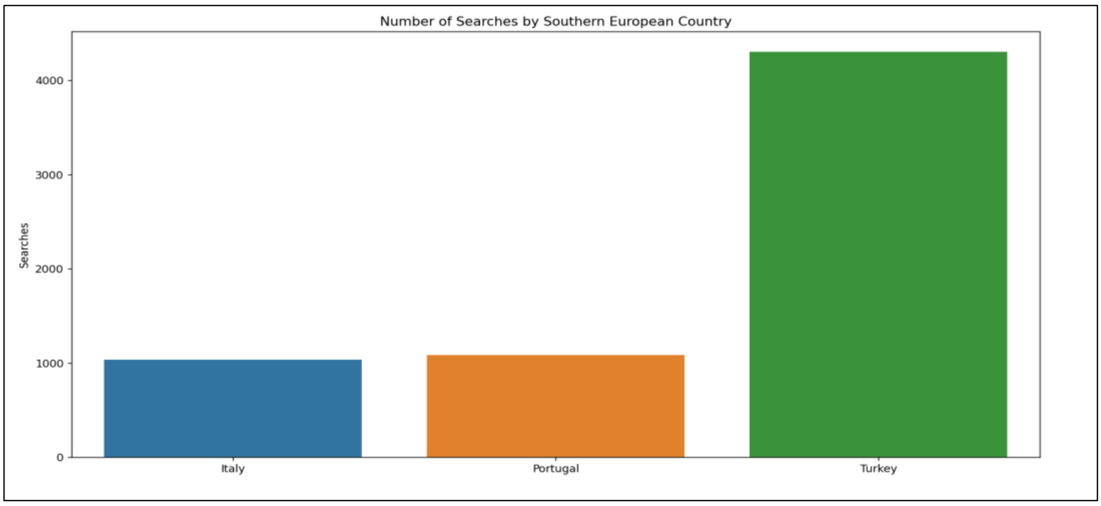


From this chart, we can see that Southern Europe has more searches than any other region. For further analysis, we can isolate all of the searches from Southern Europe.

```
# Create dataframe for Southern Europe
southEuroTrends = trends[trends["Country"].isin(southEuroCountries)]
```

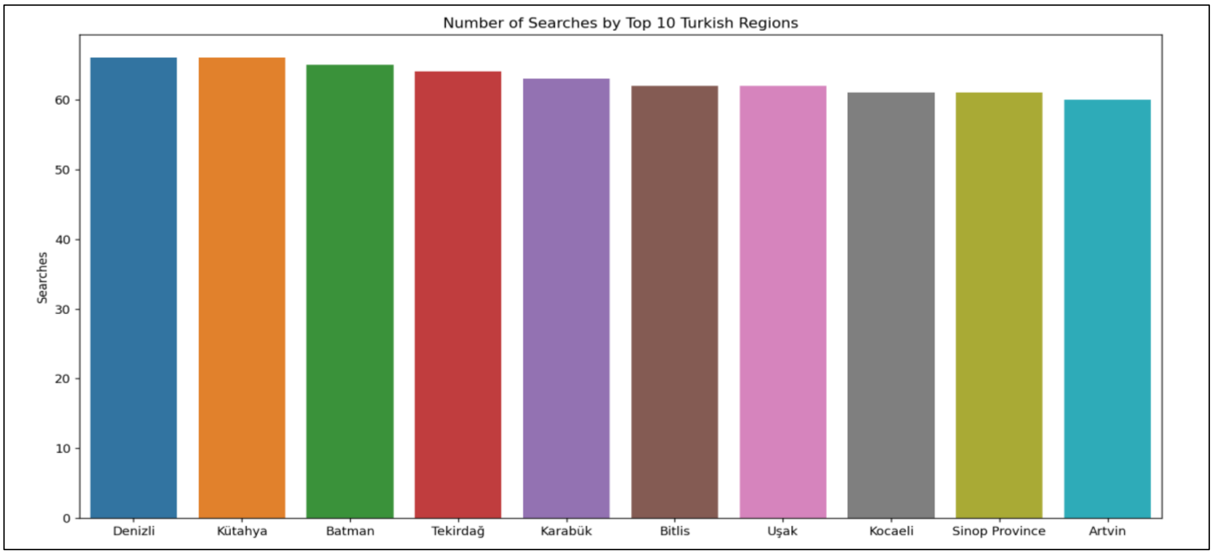
Now we can plot the number of searches against countries in Southern Europe.

```
# Plot number of searches in Southern Europe by country
southEuroTrendsCount = southEuroTrends.groupby("Country", as_index=False).count()
sns.barplot(x="Country", y="Term", data=southEuroTrendsCount)
plt.xlabel("")
plt.ylabel("Searches")
plt.title("Number of Searches by Southern European Country")
plt.show()
```



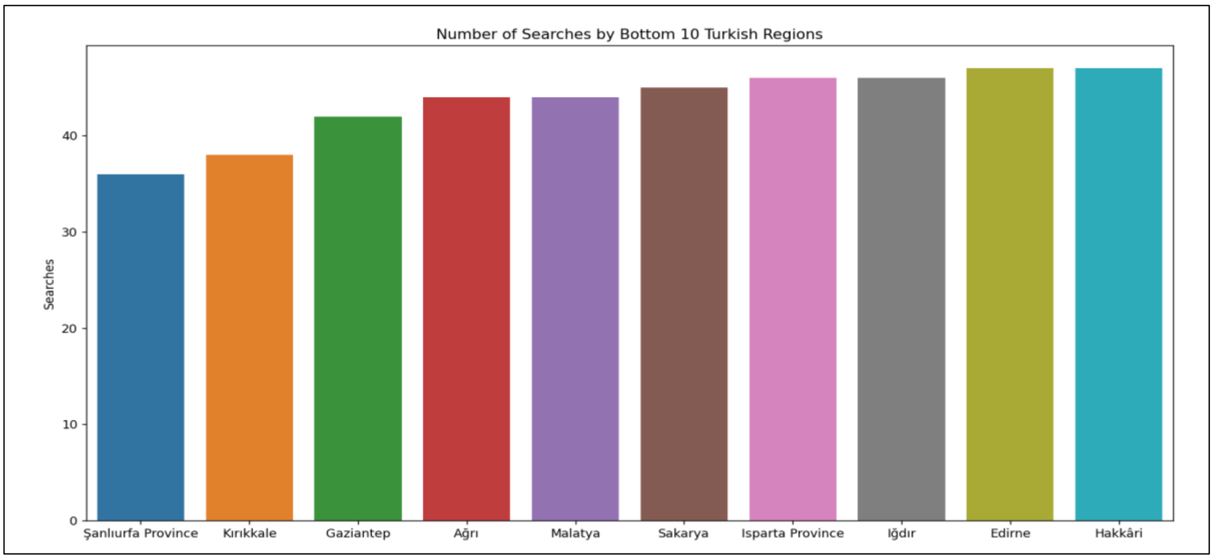
From this chart, we can see that Turkey has by far more searches than either Italy or Portugal. If we compare this with the previous chart, we can also see that Turkey has more searches than most other European regions. Since there are 81 subregions associated with searches from Turkey, it would be difficult to interpret a chart that plots all of them. To make it easier to visualize, we plot only the top 10 regions with the most searches.

```
# Plot number of searches in Turkey by top 10 regions
turkeyTrendsCount = turkeyTrends.groupby("Sub-Region", as_index=False).count()
sns.barplot(x="Sub-Region", y="Term", data=turkeyTrendsCount.nlargest(15, "Term"))
plt.xlabel("")
plt.ylabel("Searches")
plt.title("Number of Searches by Turkish Region")
plt.show()
```



From this chart, we can see that there is not a large difference in the number of trending searches from each Turkish province. For the final part of this exploration, we plot the number of searches from the bottom 10 regions.

```
# Plot number of searches in Turkey by bottom 10 regions
sns.barplot(x="Sub-Region", y="Term", data=turkeyTrendsCount.nsmallest(10, "Term"))
plt.xlabel("")
plt.ylabel("Searches")
plt.title("Number of Searches by Bottom 10 Turkish Regions")
plt.show()
```



Conclusion: -

We visited the chart for different kinds of **Data Exploration Techniques**, as we were trying to establish **Association** we focussed on the **Bivariate** ones. Started with the **Categorical - Categorical** ones, as this didn't involve **Time**, or any other **Numeric Fields** and the **Data Points** were scattered over an extremely long axis' with an enormous scale - it was not fruitful.

So, we moved on to **Bivariate Numerical - Numerical** analysis. We create **Date Intervals** and plot the **Search Terms** against it to get the **Frequencies**.

To churn out data with high frequencies we filter our **Dataset** with setting the limit of Frequency > 6. Then we drew a **Scatterplot**.

For determining correlation between **Search Term Frequency** and **Time**, we considered only one search term ('Liverpool' in this case). The **Date Interval** (that we created previously) was converted into numeric value i.e., Timestamp. Then we try to find the **Correlation** by plotting a line graph between **Search Term Frequency** and **Time Interval** (**Data Interval** converted to numeric values with **Timestamp**).

Since we get the **Correlation Coefficient Value** between **Search Term Frequency** and **Date Interval** to be 0.14, it seems less likely to be related.

To explore further we delved into **Bivariate Numerical Vs Categorical** plots.

For the **Numerical - Categorical Analysis** we used a **Bar Plot** to plot the **Number of Searches** against the **Regions of Europe** where the **Search** occurred. Since we are focusing on Europe for this exploration, we filtered our dataset for only search terms that came from European countries.

We clubbed the **European Countries** by **Region Classifiers**.

Then we created the **Bar Plot** using the **count() function** to **Aggregate Searches** by **Region**.

We saw that Southern Europe has the most Searches. So, to do a **Dipstick Study** we isolated all the **Searches** from Southern Europe.

Then we **Plotted** the **Number of Searches** against **Countries** in Southern Europe.

The **Plot** revealed that Turkey has by far more **Searches** than either Italy or Portugal, comparing this **Plot Data** with the previous **Plot Data**, we also saw that Turkey has more **Searches** than most other European regions. Turkey has **Searches** from 81 subregions, it would be difficult to interpret all of them in a single chart so to visualize better we **Plotted** only the top 10 regions with the most **Searches**.

This **Plot** revealed that there is not a large difference in the **Number of Trending Searches** from each Turkish province. For the final part of this **Exploration**, we **Plotted** the number of **Searches** from the bottom 10 regions.