Exploratory Data Analysis for Unicorn Companies Dataset

1. Introduction

This python notbook focuses on data analysis and visualization of Unicorn companies dataset recorded in March 2022 which covered

- Dataset Information
- Dataset Cleaning and exploring
- Visualization/ Visualization insights

First of all, importing necessary librabries to get started with this project.

In [1]:
```
# Importing required modules
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

2. Dataset Information

Dataset obtained via the Kaggle platform. The dataset has 13 columns and is in CSV format.
Now, loading the data for better understanding and analysis.

In [2]:
```
# Reading data set from CSV file to dataframe unicorn
unicorn = pd.read_csv("Unicorn_Companies.csv")
```

In [3]:
```
# Printing infromation about the dataframe
unicorn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1037 entries, 0 to 1036
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Company            1037 non-null   object
 1   Valuation ($B)     1037 non-null   object
 2   Date Joined        1037 non-null   object
 3   Country            1037 non-null   object
 4   City               1037 non-null   object
 5   Industry           1037 non-null   object
 6   Select Inverstors  1037 non-null   object
 7   Founded Year       1037 non-null   object
 8   Total Raised       1037 non-null   object
 9   Financial Stage    1037 non-null   object
 10  Investors Count    1037 non-null   object
 11  Deal Terms         1037 non-null   object
 12  Portfolio Exits    1037 non-null   object
dtypes: object(13)
memory usage: 105.4+ KB
```

In [4]:
```
# Printing first five records from the dataset
unicorn.head(5)
```

Out[4]:

	Company	Valuation ($B)	Date Joined	Country	City	Industry	Select Inverstors	Foun
0	Bytedance	$140	4/7/2017	China	Beijing	Artificial intelligence	Sequoia Capital China, SIG Asia Investments, S...	2
1	SpaceX	$100.3	12/1/2012	United States	Hawthorne	Other	Founders Fund, Draper Fisher Jurvetson, Rothen...	2
2	Stripe	$95	1/23/2014	United States	San Francisco	Fintech	Khosla Ventures, LowercaseCapital, capitalG	2
3	Klarna	$45.6	12/12/2011	Sweden	Stockholm	Fintech	Institutional Venture Partners, Sequoia Capita...	2
4	Epic Games	$42	10/26/2018	United States	Cary	Other	Tencent Holdings, KKR, Smash Ventures	1

3. Dataset Cleanning and exploring

After viewing the result of the info function, it is clear that every field in the dataset is an Object type, which must be changed before any operations can be performed.

Let's take a brief glance at the data before continuing

In [5]:
```
unicorn.describe()
```

Out[5]:

	Company	Valuation ($B)	Date Joined	Country	City	Industry	Select Inverstors	Founded Year	R
count	1037	1037	1037	1037	1037	1037	1037	1037	
unique	1035	200	623	46	256	33	1006	37	
top	Bolt	$1	7/13/2021	United States	San Francisco	Fintech	None	2015	
freq	2	244	9	536	145	205	17	144	

In [6]:
```
# Removing column which are not been used for analysis
unicorn = unicorn.drop(['Financial Stage','Deal Terms','Portfolio Exits'], a
```

By looking at the data it seen that in the **valuation column** dollar sign is present before text which should be removed and
same case with the **column Total Raised** where dollar sign is present and also we can see it conatin letter B and M to denote the Billion and Million value

In [7]:
```
# Removing '$' sign from values in 'Valuation ($B)'
unicorn['Valuation ($B)'] = unicorn['Valuation ($B)'].str.split('$').str[1]
unicorn['Valuation ($B)']
```

Out[7]:
```
0          140
1        100.3
2           95
3         45.6
4           42
         ...
1032         1
1033         1
1034         1
1035         1
1036         1
Name: Valuation ($B), Length: 1037, dtype: object
```

In [8]:
```
# Removing '$' and  Converting 'B' to 'M' values
test = unicorn["Total Raised"].str.split('$').str[1].astype(str)
for i, each in zip(range(len(test)), test):
    value = each[:-1]
    measure = each[-1]
    if measure == 'M':
        test[i] = float(value) * 0.001
    else:
        test[i] = value
unicorn["Total Raised"] = test

# Renaming the column to get understanding of value it contain
unicorn.rename(columns={'Total Raised' : 'Total Raised ($B)'}, inplace=True)
unicorn.head(10)
```

Out[8]:

	Company	Valuation ($B)	Date Joined	Country	City	Industry	Select Inverstors
0	Bytedance	140	4/7/2017	China	Beijing	Artificial intelligence	Sequoia Capital China, SIG Asia Investments, S...
1	SpaceX	100.3	12/1/2012	United States	Hawthorne	Other	Founders Fund, Draper Fisher Jurvetson, Rothen...
2	Stripe	95	1/23/2014	United States	San Francisco	Fintech	Khosla Ventures, LowercaseCapital, capitalG
3	Klarna	45.6	12/12/2011	Sweden	Stockholm	Fintech	Institutional Venture Partners, Sequoia Capita...
4	Epic Games	42	10/26/2018	United States	Cary	Other	Tencent Holdings, KKR, Smash Ventures
5	Canva	40	1/8/2018	Australia	Surry Hills	Internet software & services	Sequoia Capital China, Blackbird Ventures, Mat...
6	Checkout.com	40	5/2/2019	United Kingdom	London	Fintech	Tiger Global Management, Insight Partners, DST...
7	Instacart	39	12/30/2014	United States	San Francisco	Supply chain, logistics, & delivery	Khosla Ventures, Kleiner Perkins Caufield & By...
8	Databricks	38	2/5/2019	United States	San Francisco	Data management & analytics	Andreessen Horowitz, New Enterprise Associates...
9	Revolut	33	4/26/2018	United Kingdom	London	Fintech	index Ventures, DST Global, Ribbit Capital

Converting data type from object to required data type for operation

In [9]:
```
# Converting 'Valuation ($B)' from object to float type
unicorn['Valuation ($B)'] = unicorn['Valuation ($B)'].astype(float)

# Converting 'Date Time' from object to datetime type
unicorn['Date Joined'] = pd.to_datetime(unicorn['Date Joined'])


# Converting 'Founded Year' from object to datetime type
# unicorn["Founded Year"] = pd.to_datetime(unicorn["Founded Year"], format='

# Removing complete row where unicorn["Total Raised ($M)"] == 'na' value
unicorn.drop(unicorn[unicorn['Total Raised ($B)'] == 'na'].index, inplace =

# Converting 'Total Raised ($M)' from object to float type
unicorn['Total Raised ($B)'] = unicorn['Total Raised ($B)'].astype(float)
```

In [10]:
```
unicorn.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1013 entries, 0 to 1036
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Company            1013 non-null   object
 1   Valuation ($B)     1013 non-null   float64
 2   Date Joined        1013 non-null   datetime64[ns]
 3   Country            1013 non-null   object
 4   City               1013 non-null   object
 5   Industry           1013 non-null   object
 6   Select Inverstors  1013 non-null   object
 7   Founded Year       1013 non-null   object
 8   Total Raised ($B)  1013 non-null   float64
 9   Investors Count    1013 non-null   object
dtypes: datetime64[ns](1), float64(2), object(7)
memory usage: 87.1+ KB
```

In [11]:
```
unicorn.head(5)
```

Out[11]:

	Company	Valuation ($B)	Date Joined	Country	City	Industry	Select Inverstors	Founded Year
0	Bytedance	140.0	2017-04-07	China	Beijing	Artificial intelligence	Sequoia Capital China, SIG Asia Investments, S...	2012
1	SpaceX	100.3	2012-12-01	United States	Hawthorne	Other	Founders Fund, Draper Fisher Jurvetson, Rothen...	2002
2	Stripe	95.0	2014-01-23	United States	San Francisco	Fintech	Khosla Ventures, LowercaseCapital, capitalG	2010
3	Klarna	45.6	2011-12-12	Sweden	Stockholm	Fintech	Institutional Venture Partners, Sequoia Capita...	2005
4	Epic Games	42.0	2018-10-26	United States	Cary	Other	Tencent Holdings, KKR, Smash Ventures	1991

Renaming column name for better undertsnding of values

Found out some NULL/NaN/None value in dataset so removing rows whose values are NULL/NaN/None

In [12]:
```
# Renaming the column name
unicorn.rename(columns={'Select Inverstors' : 'Investors'}, inplace=True)

# Removing row where value of Founded Year is Null/NaN/None
unicorn.drop(unicorn[unicorn['Founded Year'].isna()].index, inplace = True)
```

In [13]:
```
# Creating New colums as Year from the date joined for easy calculation
unicorn['Year'] = pd.DatetimeIndex(unicorn['Date Joined']).year
```

In [14]:
```
# Removing None values from dataset where Founded year is none
unicorn.drop(unicorn[unicorn["Founded Year"] == "None"].index, inplace = Tru

# Converting data type to int from object
unicorn['Founded Year'] = unicorn['Founded Year'].astype(int)
```

In [15]:
```
# Printing the information to see data type
unicorn.info()
```

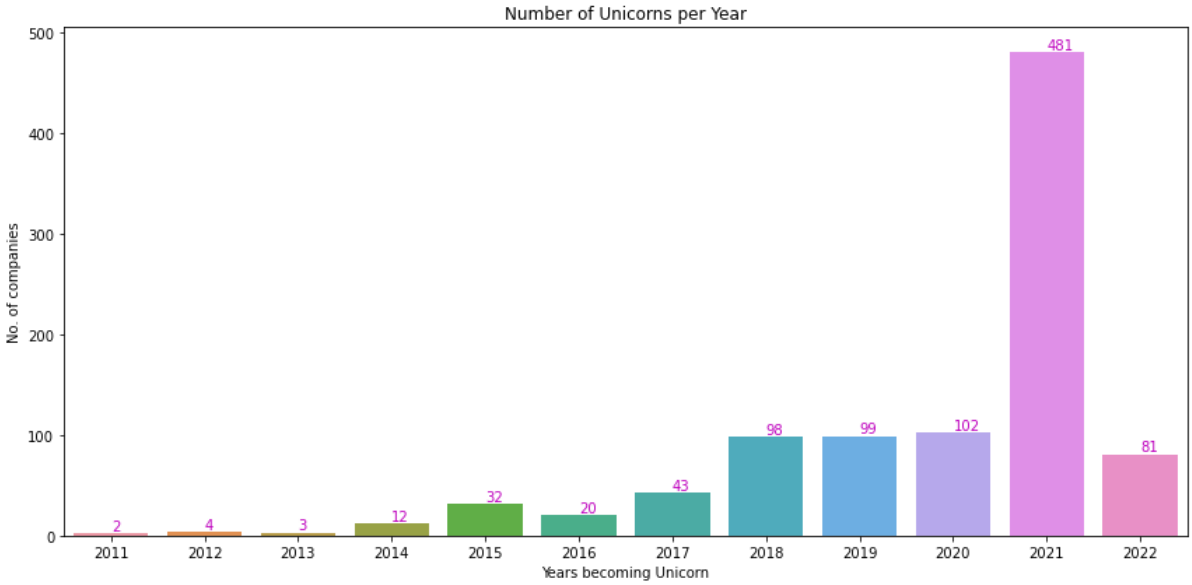
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 977 entries, 0 to 1036
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype         
---  ------             --------------  -----         
 0   Company            977 non-null    object        
 1   Valuation ($B)     977 non-null    float64       
 2   Date Joined        977 non-null    datetime64[ns]
 3   Country            977 non-null    object        
 4   City               977 non-null    object        
 5   Industry           977 non-null    object        
 6   Investors          977 non-null    object        
 7   Founded Year       977 non-null    int64         
 8   Total Raised ($B)  977 non-null    float64       
 9   Investors Count    977 non-null    object        
 10  Year               977 non-null    int64         
dtypes: datetime64[ns](1), float64(2), int64(2), object(6)
memory usage: 91.6+ KB
```

4. Data Visualization

4.1 Are unicorns a recent phenomena, and is there any trend over the past number of years in the number of unicorns being created?

Unicorn term introduced in 2013 by Aileen Lee. Current data recorded on March 2022 with 1037 companies listed in dataset as unicorn.

In [16]:
```
# Retreiving yearwise count by using groupBy clause on unicorn
yearwise = unicorn.groupby(by='Year').count().reset_index()
# Plotting graph with standard size
plt.figure(figsize=(12,6))
# Creating barchart with help of Seaborn library by providing X and Y axis c
barchart = sns.barplot(x=yearwise['Year'], y=yearwise['Company'])
# Setting lables required for reading and understanding of graph
barchart.set(xlabel = "Years becoming Unicorn", ylabel = "No. of companies")
# Adding title to graph
barchart.axes.set_title("Number of Unicorns per Year", fontsize=12)
# Iterating over the yearwise items so printing count on each bar we are plc
for i, v in enumerate(yearwise['Company'].iteritems()):
    barchart.text(i, v[1],v[1], color='m', va ='bottom')
plt.tight_layout()
plt.show()
```



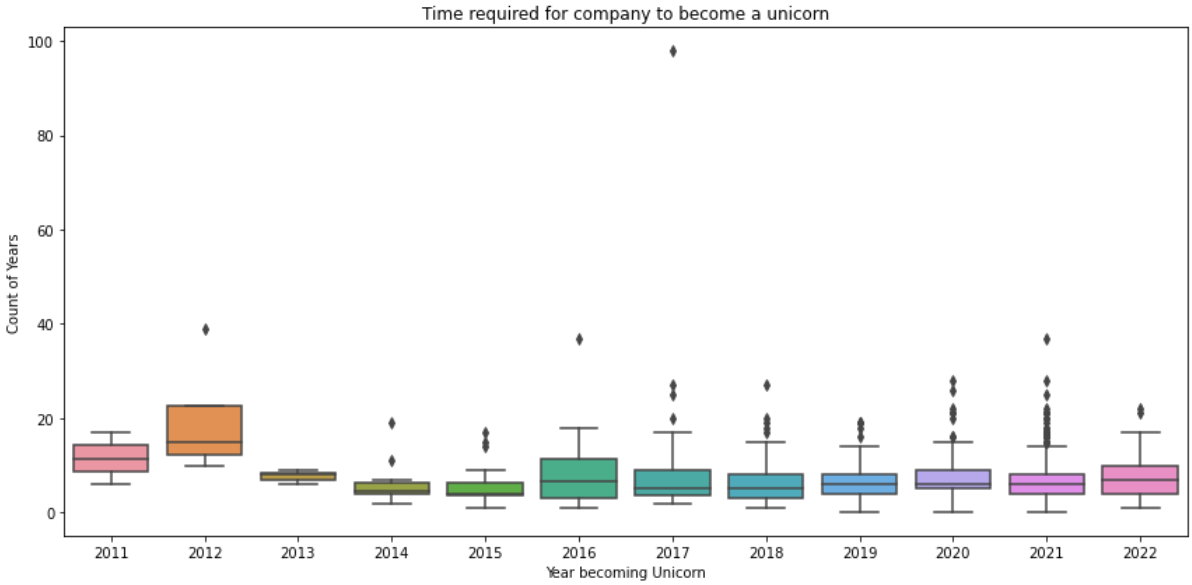
4.2 How long does it usually take for a company to become a unicorn? Has it always been this way?

By using Date Joined as Unicorn and Companu founded year we can answer the above question

In [17]:
```
# Calculating difference between Year extracted from Date Joined and Founded
unicorn['Years To Unicorn'] = (unicorn['Year'] - unicorn['Founded Year'])

# checking are there any record whose value of 'Years To Unicorn' is less th
unicorn[unicorn['Years To Unicorn']<0]
# Some records were found whose value is less than 0
# after searching on google the founded year of Yidian Zixun is 2013 so chan
unicorn.loc[unicorn['Company']=='Yidian Zixun','Founded Year']=2013
# and the years to unicron will change to 4 as (2017-2013 = 4)
unicorn.loc[unicorn['Company']=='Yidian Zixun','Years To Unicorn']=4

# Plotting graph with standard size
plt.figure(figsize=(12,6))
# Creating boxplot with help of Seaborn library by providing X and Y data va
boxplot = sns.boxplot(x=unicorn['Year'], y=unicorn['Years To Unicorn'])
# Setting lables required for reading and understanding of graph
boxplot.set(xlabel = "Year becoming Unicorn", ylabel = "Count of Years")
# Adding title to graph
boxplot.axes.set_title("Time required for company to become a unicorn", font
plt.tight_layout()
plt.show()
```



4.3 Which top 10 companies tunred into unicorn the fastest?

To display the graph and provide an answer, we need the top 10 companies and the number of years it takes for them to become unicorns.

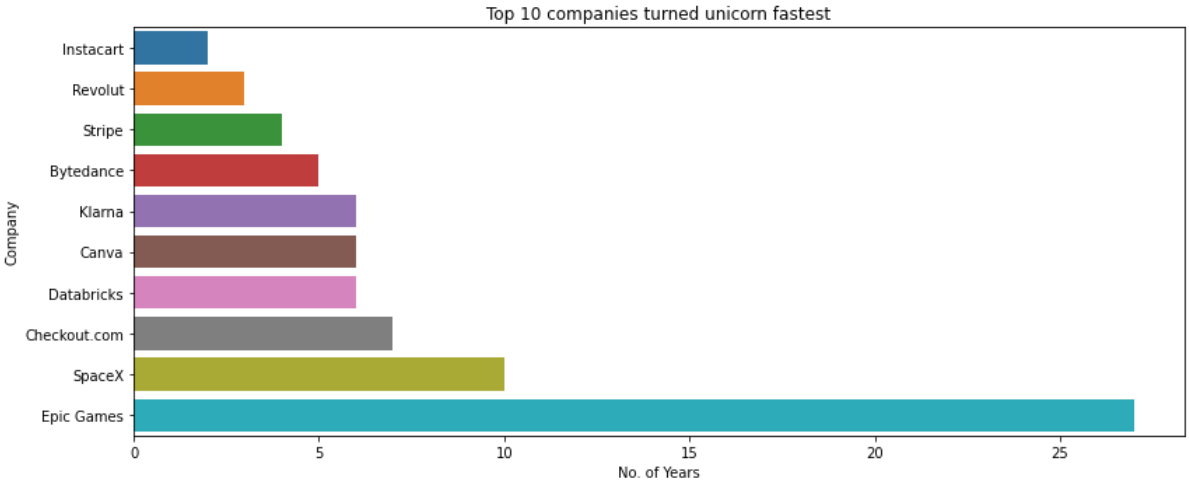
In [18]:
```
# Reteriving Top companies on basis of Valuation
top10Companies = unicorn.sort_values(by='Valuation ($B)', ascending=False).h
# sorting top 10 companies on basis of number of years required
top10Companies = top10Companies.sort_values(by='Years To Unicorn').reset_ind
top10Companies
```

Out[18]:

	index	Company	Valuation ($B)	Date Joined	Country	City	Industry	Investo
0	7	Instacart	39.0	2014-12-30	United States	San Francisco	Supply chain, logistics, & delivery	Khosla Venture Kleiner Perki Caufield & By
1	9	Revolut	33.0	2018-04-26	United Kingdom	London	Fintech	index Venture DST Global, Ribb Capit
2	2	Stripe	95.0	2014-01-23	United States	San Francisco	Fintech	Khosla Venture LowercaseCapit capita
3	0	Bytedance	140.0	2017-04-07	China	Beijing	Artificial intelligence	Sequoia Capit China, SIG As Investments, S
4	3	Klarna	45.6	2011-12-12	Sweden	Stockholm	Fintech	Institutior Venture Partner Sequoia Capita
5	5	Canva	40.0	2018-01-08	Australia	Surry Hills	Internet software & services	Sequoia Capit China, Blackbi Ventures, Mat
6	8	Databricks	38.0	2019-02-05	United States	San Francisco	Data management & analytics	Andreesse Horowitz, Ne Enterpri Associates
7	6	Checkout.com	40.0	2019-05-02	United Kingdom	London	Fintech	Tiger Glob Managemer Insight Partner DST
8	1	SpaceX	100.3	2012-12-01	United States	Hawthorne	Other	Founders Fun Draper Fish Jurvetso Rothen
9	4	Epic Games	42.0	2018-10-26	United States	Cary	Other	Tencent Holding KKR, Sma Ventur

In [19]:

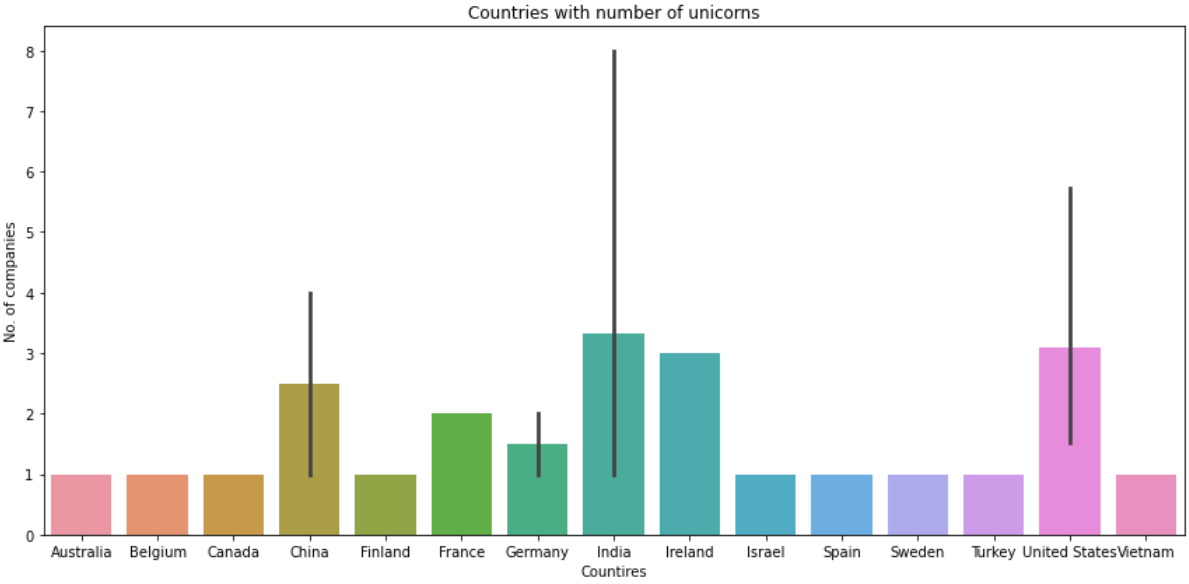
```
# Plotting graph with standard size
plt.figure(figsize=(12,5))
# Creating barchart with help of Seaborn library by providing X and Y data v
barchart = sns.barplot(x = 'Years To Unicorn', y = 'Company', data=top10Comp
# Setting lables required for reading and understanding of graph
barchart.set(xlabel ="No. of Years" , ylabel = "Company")
# Adding title to graph
barchart.axes.set_title("Top 10 companies turned unicorn fastest", fontsize=
plt.tight_layout()
plt.show()
```



4.4 Which countries have the most unicorns Are there any countries that appear to be industry as Internet software & services?

required to be find countries with the industry matching to Internet Software & services and then for that we need to calculate count of companies

In [20]:
```
localdf = unicorn.groupby(['Country','Industry','City'])['Company'].count().
localdf = localdf[localdf['Industry'] == 'Internet software & services']
# Plotting graph with standard size
plt.figure(figsize=(12,6))
# Creating barchart with help of Seaborn library by providing X and Y axis d
barchart = sns.barplot(x=localdf['Country'], y=localdf['Company'])
# Setting lables required for reading and understanding of graph
barchart.set(xlabel = "Countires", ylabel = "No. of companies")
# Adding title to graph
barchart.axes.set_title("Countries with number of unicorns", fontsize=12)
plt.tight_layout()
plt.show()
```



4.5 Yearly valuation of top 5 countries

In order to calculate the data, it is necessary to first determine the top 5 countries with the highest valuations and then retrieve the data for those countries to calculate the valuations by year.

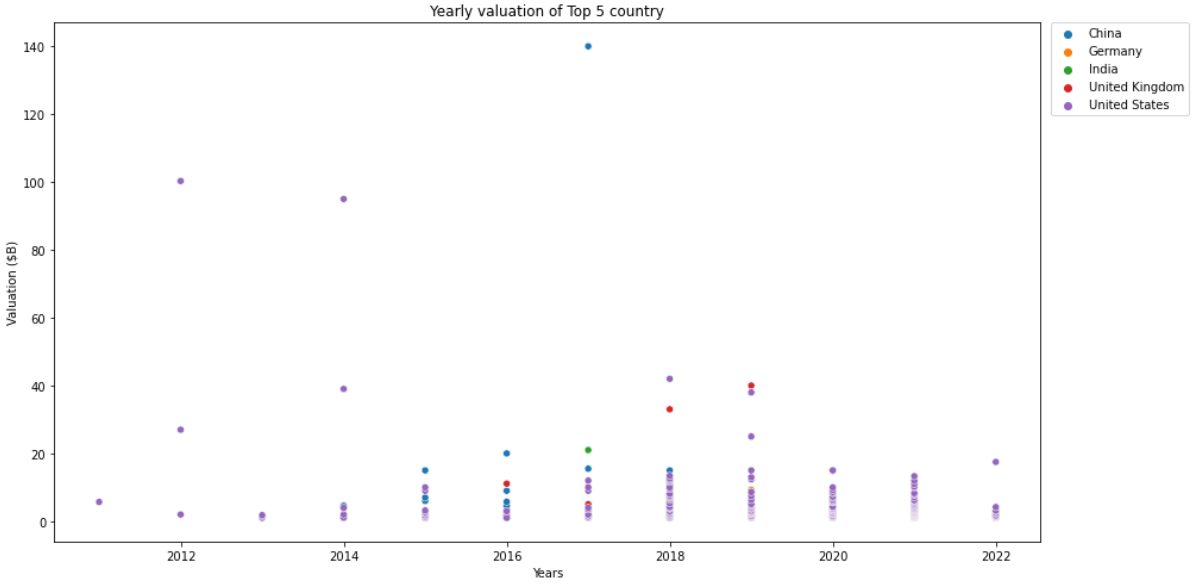
In [21]:
```
# Reteriving the top 5 countries on basis of Valuation
top5Country = unicorn.groupby(['Country'])['Valuation ($B)'].sum().reset_ind
# creating list of to 5 countries
countrylist = top5Country['Country'].tolist()
# Reteriving year country and valuation data
localdf = unicorn.groupby(['Country','Year','Valuation ($B)'])['Company'].co
# finding out Reterived data contain have countries from top 5 country list
localdf = localdf[localdf['Country'].isin(countrylist)]
# printing data to ensure it is correct
localdf
```

Out[21]:

	Country	Year	Valuation ($B)	Company
42	China	2014	1.40	1
43	China	2014	4.58	1
44	China	2015	1.00	9
45	China	2015	1.05	1
46	China	2015	1.50	1
...	...	...	...	...
592	United States	2022	2.50	2
593	United States	2022	3.00	1
594	United States	2022	3.20	1
595	United States	2022	4.20	1
596	United States	2022	17.50	1

425 rows × 4 columns

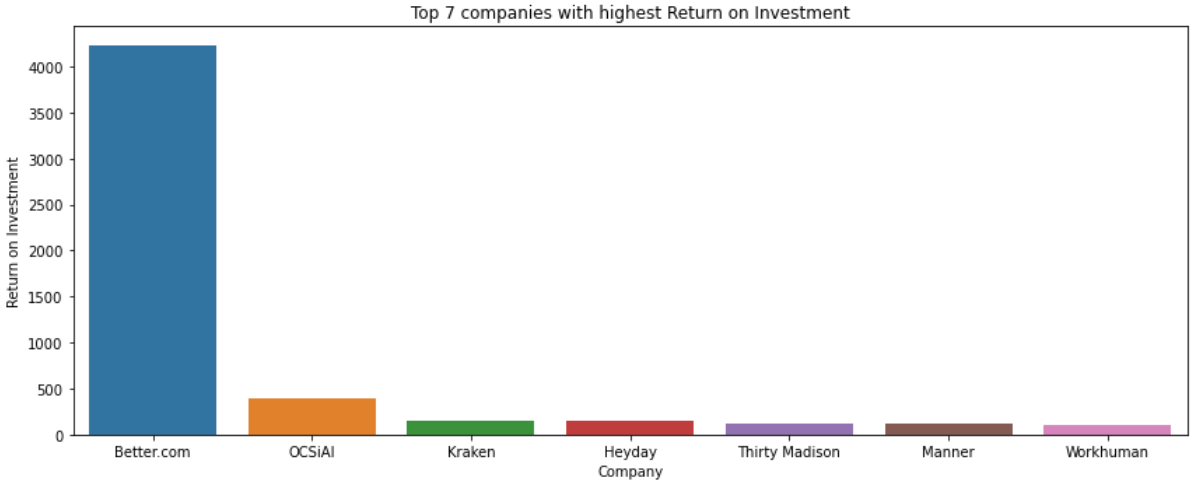
In [22]:
```
# Plotting graph with standard size
plt.figure(figsize=(14,7))
# Creating scatter plot with help of Seaborn library by providing X and Y da
scatterplotchart = sns.scatterplot(data = localdf, x = "Year", y = "Valuatic
# Setting lables required for reading and understanding of graph
scatterplotchart.set(xlabel ="Years" , ylabel = "Valuation ($B)")
# Adding title to graph
scatterplotchart.axes.set_title("Yearly valuation of Top 5 country", fontsiz
# Adjusting the legend for graph
plt.legend(bbox_to_anchor=(1.15, 1), borderaxespad=0)
plt.tight_layout()
plt.show()
```



4.6 Which unicorn companies have had the biggest return on investment ?

(Showing result for top 7 companies)

In [23]:
```
unicorn['ROI'] = (unicorn['Valuation ($B)'] - unicorn['Total Raised ($B)'])
top7roi = unicorn.sort_values(by='ROI', ascending=False).head(7)
# Plotting graph with standard size
plt.figure(figsize=(12,5))
# Creating barchart with help of Seaborn library by providing X and Y data v
barchart = sns.barplot(x = 'Company', y = 'ROI', data=top7roi)
# Setting lables required for reading and understanding of graph
barchart.set(xlabel ="Company" , ylabel = "Return on Investment")
# Adding title to graph
barchart.axes.set_title("Top 7 companies with highest Return on Investment",
plt.tight_layout()
plt.show()
```



In [ ]: