Top 2000 Compagnies in the world based on Forbes

In this project we are going to analyse the structure, the performance of the Top 2000 compagnies in the world

Financial metrics (Sales , Profit , Assets , Market Value) are in billions of dollars.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from warnings import  filterwarnings
filterwarnings('ignore')
```

```
df = pd.read_csv('Largest-Companies.csv')
df.head()
```

	Rank int64	Name object	Sales float64	Profit float64	Assets float64	Market Value flo...	Industry object	F
0	1	JPMorgan Chase	252.9	50	4090.7	588.1	Banking and Fina...	
1	2	Berkshire Hathaw...	369	73.4	1070	899.1	Conglomerate	
2	3	Saudi Arabian Oil ...	489.1	116.9	661.5	1919.3	Construction- Ch...	
3	4	ICBC	223.8	50.4	6586	215.2	Banking and Fina...	
4	5	Bank of America	183.3	25	3273.8	307.3	Banking and Fina...	

5 rows, 12 cols 10 / page Page 1 of 1

```
df.shape
```

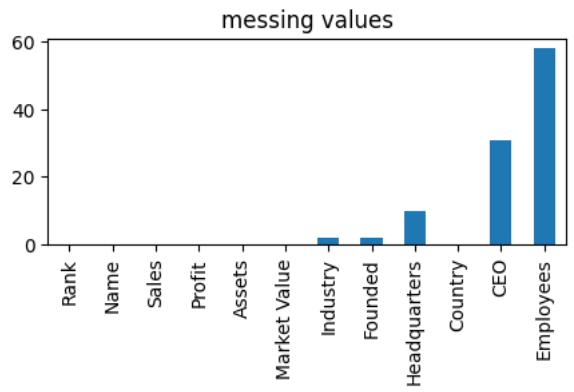
```
(2001, 12)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2001 entries, 0 to 2000
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Rank          2001 non-null   int64
 1   Name          2001 non-null   object
 2   Sales         2001 non-null   float64
 3   Profit        2001 non-null   float64
 4   Assets        2001 non-null   float64
 5   Market Value  2001 non-null   float64
 6   Industry      1999 non-null   object
 7   Founded       1999 non-null   float64
 8   Headquarters  1991 non-null   object
 9   Country       2001 non-null   object
 10  CEO           1970 non-null   object
 11  Employees     1943 non-null   float64
dtypes: float64(6), int64(1), object(5)
memory usage: 187.7+ KB
```

```
plt.figure(figsize=(5,2))
df.isna().sum().plot(kind='bar')
plt.title('messing values')
```

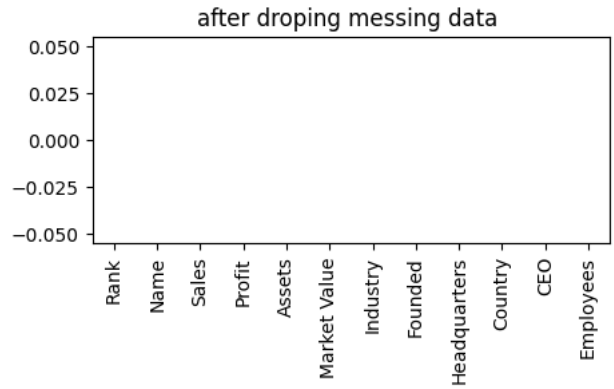
```
Text(0.5, 1.0, 'messing values')
```




```
df.dropna(axis=0,inplace=True)
```

```
plt.figure(figsize=(5,2))
df.isna().sum().plot(kind='bar')
plt.title('after droping messing data ')
```

```
Text(0.5, 1.0, 'after droping messing data ')
```




```
[col for col in df.columns if df[col].dtype == 'object' ]
```

```
['Name', 'Industry', 'Headquarters', 'Country', 'CEO']
```

```
def map_category(Industry):
    if pd.notnull(Industry):
        if "Banking" in Industry or "Financial Services" in Industry:
            return "Banking_and_Financial_Services"
        elif "Software" in Industry or "Technology" in Industry or "Semiconductors" in Industry:
            return "Technology & Software"
        elif "Retail" in Industry or "Wholesale" in Industry or "Consumer" in Industry:
            return "Retail & Consumer Goods"
        elif "Construction" in Industry or "Real Estate" in Industry or "Materials" in Industry:
            return "Construction & Real Estate"
        elif "Health" in Industry or "Medical" in Industry or "Pharma" in Industry:
            return "Healthcare & Pharmaceuticals"
        elif "Food" in Industry or "Drink" in Industry or "Restaurant" in Industry or "Hotel" in Industry:
            return "Food & Hospitality"
        elif "Media" in Industry or "Advertising" in Industry:
            return "Media & Entertainment"
        elif "Transport" in Industry or "Auto" in Industry or "Logistics" in Industry:
            return "Transportation & Logistics"
        elif "Oil" in Industry or "Gas" in Industry or "Utilities" in Industry:
            return "Energy & Utilities"
        elif "Service" in Industry or "Business" in Industry or "Professional" in Industry:
            return "Professional Services"
        elif "Conglomerate" in Industry:
            return "Conglomerates"
        else:
            return "Miscellaneous"
    return "Miscellaneous"
df['domain'] = df['Industry'].apply(map_category)
```

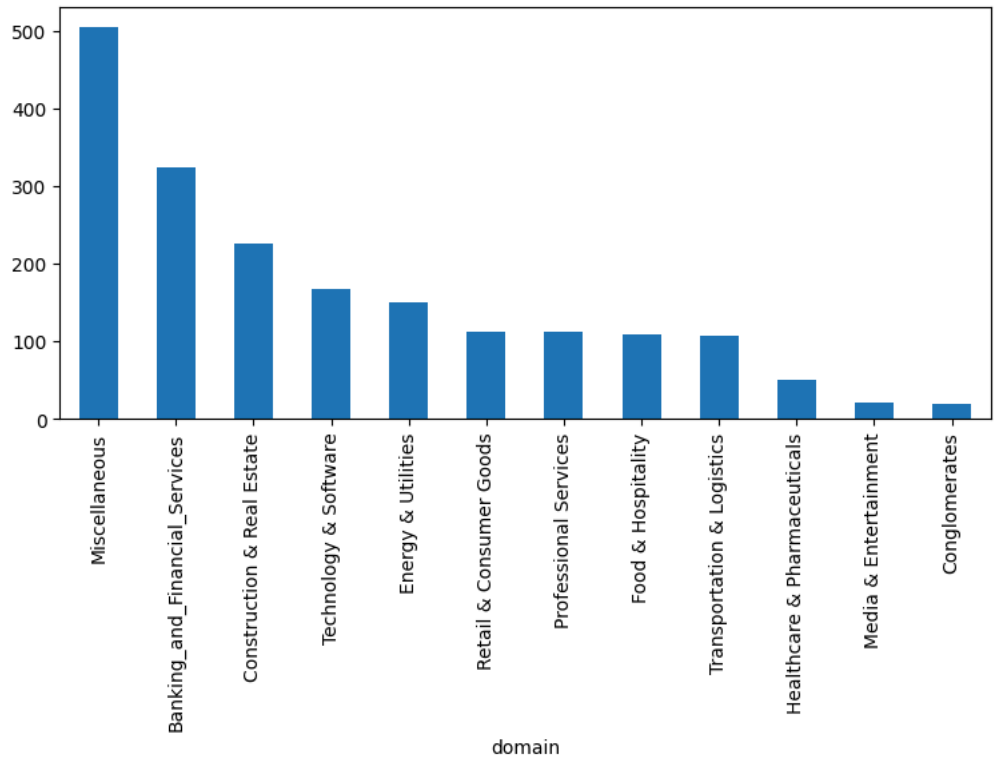
```
df.head(4)
```

	Rank int64	Name object	Sales float64	Profit float64	Assets float64	Market Value flo...	Industry object	F
0	1	JPMorgan Chase	252.9	50	4090.7	588.1	Banking and Fina...	
1	2	Berkshire Hathaw...	369	73.4	1070	899.1	Conglomerate	
2	3	Saudi Arabian Oil ...	489.1	116.9	661.5	1919.3	Construction- Ch...	
3	4	ICBC	223.8	50.4	6586	215.2	Banking and Fina...	

4 rows, 13 cols 10 / page Page 1 of 1

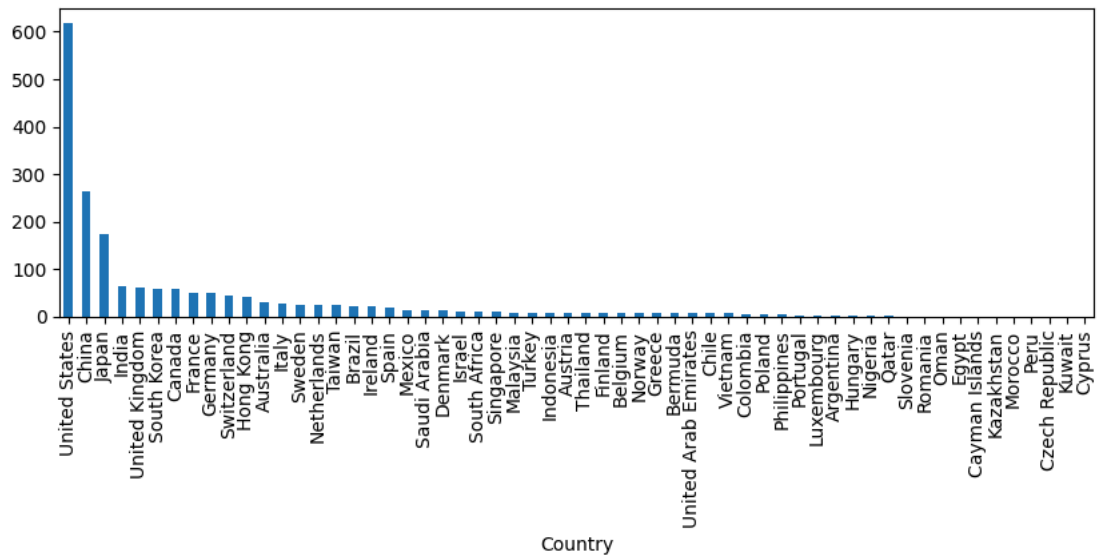
```
plt.figure(figsize=(9,4))
df.domain.value_counts().plot(kind='bar')
```

```
<AxesSubplot: xlabel='domain'>
```



```
plt.figure(figsize=(10,3))
df.Country.value_counts().plot(kind='bar')
```

```
<AxesSubplot: xlabel='Country'>
```




```
df.describe()
```

	Rank float64	Sales float64	Profit float64	Assets float64	Market Value flo...	Founded float64	Employees float64
cou...	1906	1906	1906	1906	1906	1906	1906
me...	985.1495278	26.23691186	2.313138458	122.2025184	45.51028856	1966.165792	48926.71354
std	577.6024682	46.84353633	6.318784238	397.2218173	152.737721	417.5834891	90106.4734
min	1	0.165	-17.9	1.3	0.007	1472	11
25%	484.25	6.1	0.5069	15.5	8.3	1925	8571.75
50%	976	13.1	0.9587	33.8	17.8	1972	21987.5
75%	1479.75	25.7	2.2	78.6	39	1996	52725
max	2001	657.3	116.9	6586	3123.1	20047	1600000

8 rows, 7 cols 10 / page « ‹ Page 1 of 1 › »

```
print(df.columns)
```

```
Index(['Rank', 'Name', 'Sales', 'Profit', 'Assets', 'Market Value', 'Industry',
       'Founded', 'Headquarters', 'Country', 'CEO', 'Employees', 'domain'],
      dtype='object')
```

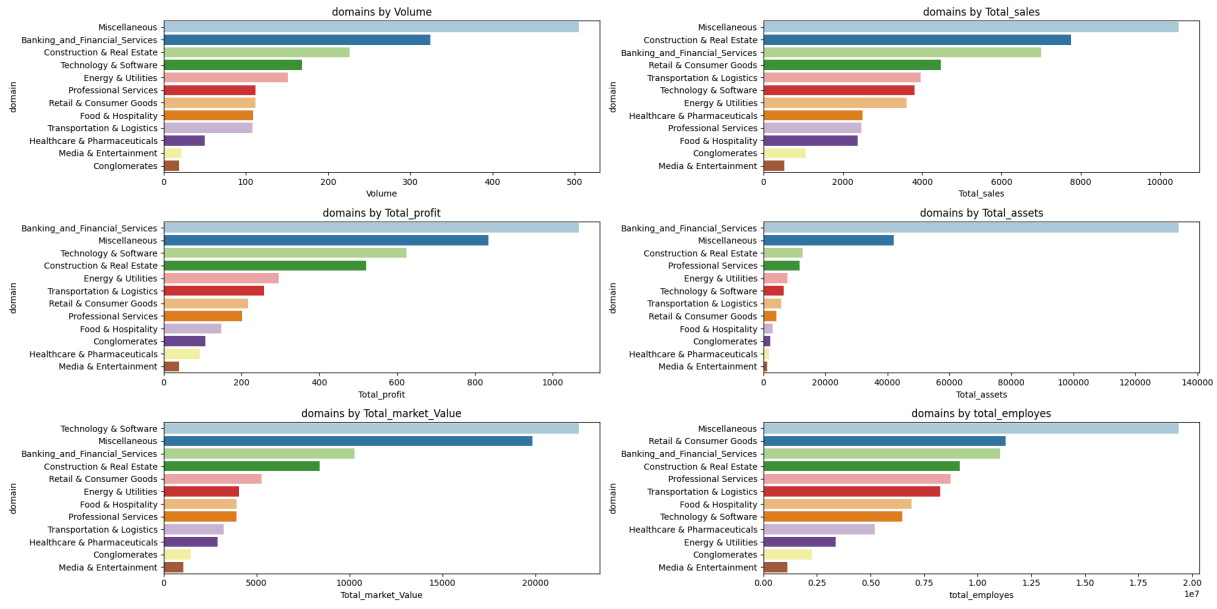
```
compagnies_domain = df.groupby('domain').agg(
    Volume = ('Name','count'),
    Total_sales = ('Sales','sum'),
    avg_sales = ('Sales','mean'),
    Total_profit = ('Profit','sum'),
    avg_profit = ('Profit','mean'),
    Total_assets = ('Assets','sum'),
    avg_assets = ('Assets','mean'),
    Total_market_Value = ('Market Value','sum'),
    total_employes = ('Employees','sum')
    )
```

```
compagnies_domain
```

	Volume int64 19 - 505	**Total_sales** float64 530.0 - 10473.591	**avg_sales** float64 20.73978415841...	**Total_profit** float... 39.7062 - 1067.584	**avg_profit** float64 1.357368807339...	**Total_assets** floa... 1141.7 - 134041.5	**avg_assets** float64 27.99724770642...	T 1
Ba...	324	6995.028	21.58959259	1067.584	3.295012346	134041.5	413.7083333	
Co...	19	1073.1	56.47894737	108.1666	5.692978947	2280.1	120.0052632	
Co...	226	7755.762	34.31753097	521.4241	2.307186283	12643.9	55.94646018	
En...	151	3606.966	23.88719205	295.8752	1.959438411	7717	51.10596026	
Foo...	109	2371.557	21.75740367	147.9532	1.357368807	3051.7	27.99724771	
He...	50	2496.7	49.934	92.6493	1.852986	1747.5	34.95	
Me...	22	530	24.09090909	39.7062	1.804827273	1141.7	51.89545455	
Mis...	505	10473.591	20.73978416	834.9094	1.653285941	42052.4	83.27207921	
Pro...	112	2464.6	22.00535714	201.3442	1.797716071	11730.1	104.7330357	
Ret...	112	4469.2	39.90357143	216.771	1.935455357	4147.1	37.02767857	

12 rows, 9 cols 10 / page Page 1 of 2

```
def plot (data):
    fig,ax = plt.subplots(3,2,figsize=(20,10))
    ax = ax.flatten()
    for i, col in enumerate(['Volume', 'Total_sales','Total_profit'
       ,'Total_assets', 'Total_market_Value',
       'total_employes']):
        data = data.sort_values(by=col,ascending=False)
        sns.barplot(data,y=data.index,x=col,hue='domain',palette='Paired',ax=ax[i] ,legend=False)
        ax[i].set_title(f'domains by {col}',size=12)
    plt.tight_layout()
plot(compagnies_domain)
```



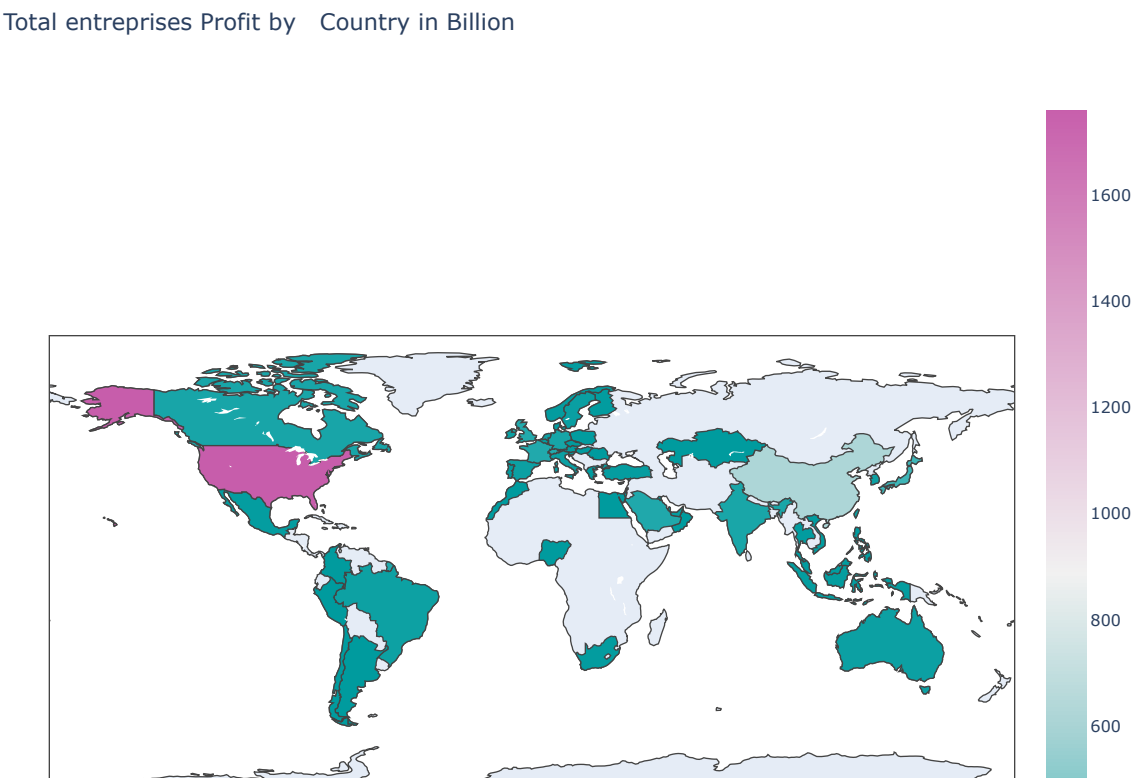
```
compagnies_Country = df.groupby('Country').agg(
    Volume = ('Name','count'),
    Total_sales = ('Sales','sum'),
    avg_sales = ('Sales','mean'),
    Total_profit = ('Profit','sum'),
    avg_profit = ('Profit','mean'),
    Total_assets = ('Assets','sum'),
    avg_assets = ('Assets','mean'),
    Total_market_Value = ('Market Value','sum'),
    total_employes = ('Employees','sum')
    )
compagnies_Country.head()
```

	Volume int64	**Total_sales** float64	**avg_sales** float64	**Total_profit** float...	**avg_profit** float64	**Total_assets** floa...	**avg_assets** float64	T
Arg...	2	35.3	17.65	2.1	1.05	31.8	15.9	
Au...	31	546.823	17.63945161	53.0804	1.712270968	3966	127.9354839	
Au...	9	153	17	12.9197	1.435522222	845.8	93.97777778	
Bel...	9	157.2	17.46666667	15.2327	1.692522222	770	85.55555556	
Ber...	8	97.1	12.1375	14.101	1.762625	349.9	43.7375	

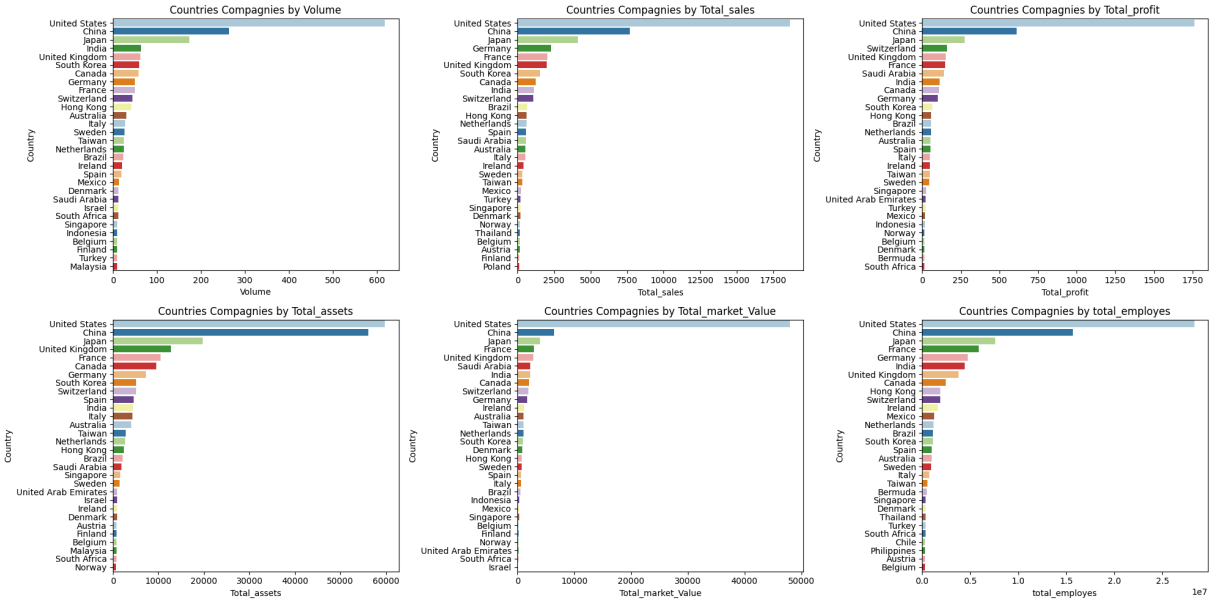
5 rows, 9 cols 10 / page Page 1 of 1

```
import plotly.graph_objects as go
fig = go.Figure()
fig.add_trace(go.Choropleth(
    locations = compagnies_Country.index,
    locationmode = 'country names',
    z=compagnies_Country['Total_profit'],
    colorscale='tropic',
    text = compagnies_Country['Total_profit'],
    hoverinfo = 'location+z',

))
fig.update_layout(
    title="  Total entreprises Profit by   Country in Billion",
    height = 900
)
fig.show()
```



```
def plot (data):
    fig,ax = plt.subplots(2,3,figsize=(20,10))
    ax = ax.flatten()
    for i, col in enumerate(['Volume', 'Total_sales','Total_profit'
       ,'Total_assets', 'Total_market_Value',
       'total_employes']):
        data = data.sort_values(by=col,ascending=False)
        sns.barplot(data.iloc[:30,],y=data.iloc[:30,].index,x=col,hue='Country',palette='Paired',ax=ax[i] ,legend=False)
        ax[i].set_title(f'Countries Compagnies by {col}',size=12)
    plt.tight_layout()
plot(compagnies_Country)
```




```
df['Profit_Ratio'] = df['Profit']/df['Sales']
df['retun_of_assets'] = df['Profit']/df['Assets']
df['market_to_assets'] = df['Market Value']/df['Assets']
df['sales per employee'] = df['Sales']/df['Employees']
df['Profit per employe'] = df['Profit']/df['Employees']
df['assets_utilisation'] = df['Sales']/df['Assets']
```

```
df_top5 = df[df.Rank <= 5]
df_top5.columns
```

```
Index(['Rank', 'Name', 'Sales', 'Profit', 'Assets', 'Market Value', 'Industry',
       'Founded', 'Headquarters', 'Country', 'CEO', 'Employees', 'domain',
       'Profit_Ratio', 'retun_of_assets', 'market_to_assets',
       'sales per employee', 'Profit per employe', 'assets_utilisation'],
      dtype='object')
```

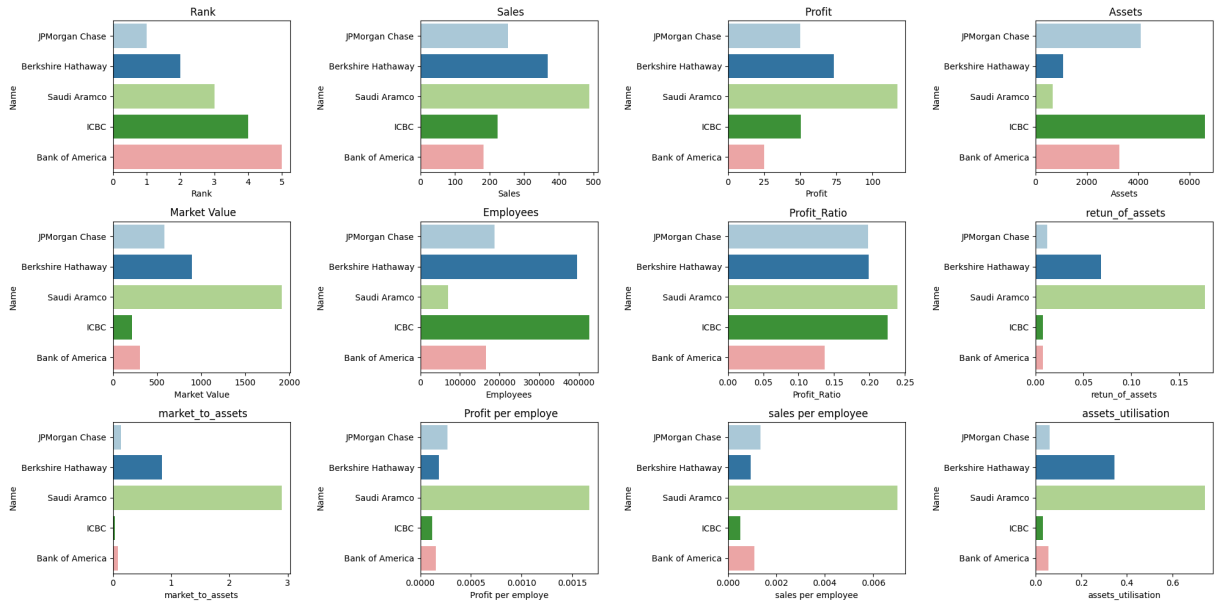
```
cols = ['Rank','Sales', 'Profit', 'Assets', 'Market Value',
         'Employees',
       'Profit_Ratio', 'retun_of_assets', 'market_to_assets',
       'Profit per employe','sales per employee', 'assets_utilisation']
```

```
from IPython.display import Markdown, display
```

```
df_top5.at[2, 'Name'] = 'Saudi Aramco'
```

```
def plot (data):
    fig,ax = plt.subplots(3,4,figsize=(20,10))
    ax = ax.flatten()
    for i, col in enumerate(cols):
        sns.barplot(data,y='Name',x=col,hue='Name',palette='Paired',ax=ax[i] ,legend=False)
        ax[i].set_title(f' {col}',size=12)
    for j in range(len(cols), len(ax)):
        fig.delaxes(ax[j])

    plt.tight_layout()
plot(df_top5)
```




```
df_num = df[['Sales', 'Profit', 'Assets', 'Market Value',
       'Headquarters', 'Employees', 'domain',
       'Profit_Ratio', 'retun_of_assets', 'market_to_assets',
       'sales per employee', 'Profit per employe', 'assets_utilisation']]
df_num.head()
```

	Sales float64	**Profit** float64	**Assets** float64	**Market Value** flo...	**Headquarters** ob...	**Employees** float64	**domain** object	P
0	252.9	50	4090.7	588.1	New York- New Y...	186751	Banking_and_Fin...	
1	369	73.4	1070	899.1	Omaha- Nebraska	396500	Conglomerates	
2	489.1	116.9	661.5	1919.3	Dhahran	70000	Construction & R...	
3	223.8	50.4	6586	215.2	Beijing	427587	Banking_and_Fin...	
4	183.3	25	3273.8	307.3	Charlotte- North ...	166140	Banking_and_Fin...	

5 rows, 13 cols 10 / page Page 1 of 1

```
dummies = pd.get_dummies(df['domain'])
dummies = dummies.map(lambda x: 1 if x == True else 0 )
dummies
```

	Banking_and_Fin... 0 - 1	Conglomerates i... 0 - 1	Construction & R... 0 - 1	Energy & Utilities i.. 0 - 1	Food & Hospitality i 0 - 1	Healthcare & Pha... 0 - 1	Media & Entertai... 0 - 1	M 0
0	1	0	0	0	0	0	0	
1	0	1	0	0	0	0	0	
2	0	0	1	0	0	0	0	
3	1	0	0	0	0	0	0	
4	1	0	0	0	0	0	0	
5	0	0	0	0	0	0	0	
6	1	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	
8	1	0	0	0	0	0	0	
9	0	0	0	0	0	0	0	

1906 rows, 12 cols 10 / page Page 1 of 191

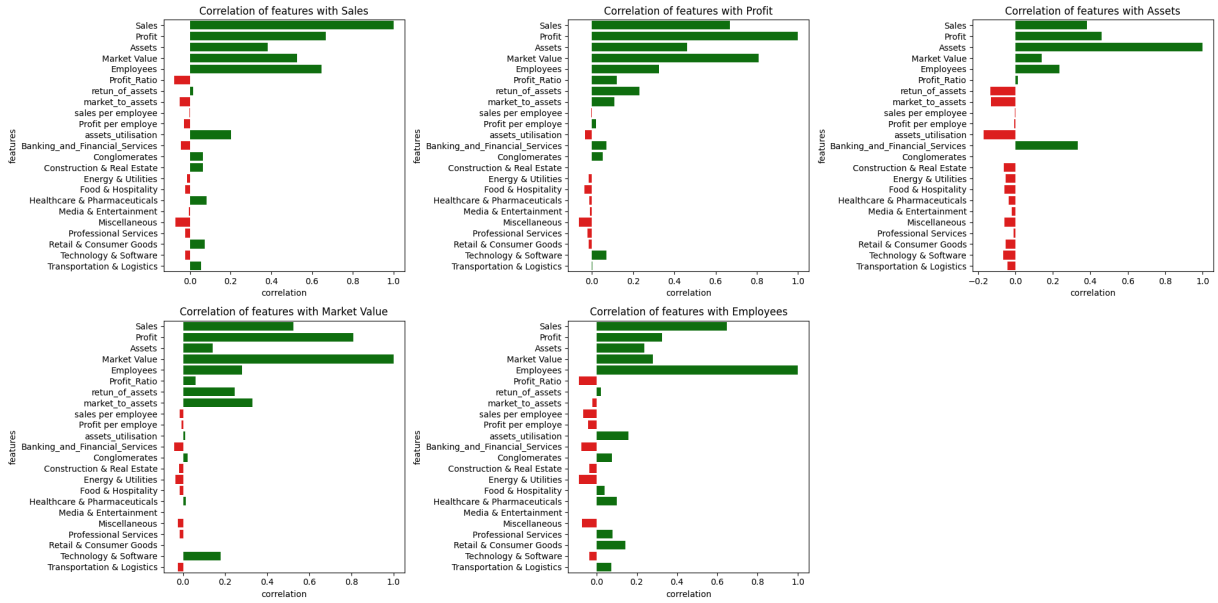
```
dummies.columns
```

```
Index(['Banking_and_Financial_Services', 'Conglomerates',
       'Construction & Real Estate', 'Energy & Utilities',
       'Food & Hospitality', 'Healthcare & Pharmaceuticals',
       'Media & Entertainment', 'Miscellaneous', 'Professional Services',
       'Retail & Consumer Goods', 'Technology & Software',
       'Transportation & Logistics'],
      dtype='object')
```

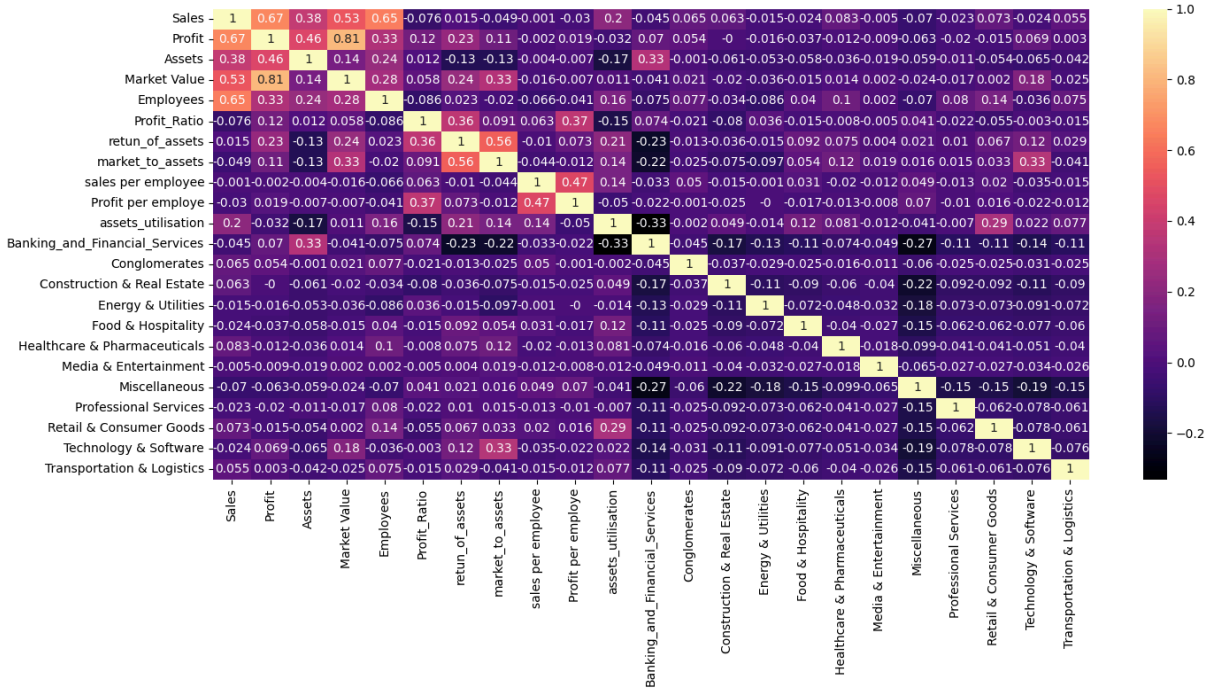
```
df_num = pd.concat([df_num,dummies],axis=1)
df_num.drop('domain',axis=1,inplace=True)
df_num.drop('Headquarters',axis=1,inplace=True)
```

```
def corr_plot(data):
    fig,ax = plt.subplots(2,3,figsize=(20,10))
    ax = ax.flatten()
    cols = ['Sales', 'Profit', 'Assets', 'Market Value', 'Employees']

    for i , col in enumerate(cols):
        corr = pd.DataFrame(data.corrwith(data[col]),columns=['correlation'])
        sns.barplot(corr,y=corr.index,x='correlation',legend=False,ax=ax[i],palette=['red' if x <= 0 else 'green' for x i
        ax[i].set_title(f'Correlation of features with {col}')
        ax[i].set_ylabel('features')
    for j in range(len(cols), len(ax)):
        fig.delaxes(ax[j])
    plt.tight_layout()
corr_plot(df_num)
```



```
plt.figure(figsize=(15,8))
sns.heatmap(round(df_num.corr(),3),cmap='magma',annot=True)
plt.tight_layout()
```




```
from sklearn.model_selection import  train_test_split, cross_val_score
from sklearn.preprocessing import  StandardScaler
from sklearn.ensemble import  RandomForestRegressor
from xgboost import  XGBRegressor
from sklearn.svm import  SVR
from sklearn.metrics import  mean_absolute_percentage_error,mean_squared_error
from sklearn.compose import ColumnTransformer
```

```
col_contunious = list(df_num.iloc[:,:11].columns)
col_binary = list(df_num.iloc[:,11:].columns)
print(col_contunious),print(col_binary)
```

```
['Sales', 'Profit', 'Assets', 'Market Value', 'Employees', 'Profit_Ratio', 'retun_of_assets', 'market_to_assets', 'sales per employee', 'Prof
['Banking_and_Financial_Services', 'Conglomerates', 'Construction & Real Estate', 'Energy & Utilities', 'Food & Hospitality', 'Healthcare & P
```

```
(None, None)
```

```
Transformer = ColumnTransformer(
    transformers = [('continuous',StandardScaler(),col_contunious),
                    ('binary','passthrough',col_binary)]
                    )
df_scaled = Transformer.fit_transform(df_num)
df_scaled = pd.DataFrame(df_scaled,columns=col_contunious+col_binary)
df_scaled.head()
```

	Sales float64	Profit float64	Assets float64	Market Value flo...	Employees float64	Profit_Ratio float...	retun_of_assets f...	n
0	4.839996929	7.548821789	9.993255083	3.553359976	1.52997283	0.2619406443	-0.5397347062	
1	7.319110965	11.25303767	2.386692204	5.59006456	3.858374298	0.2663283572	0.3406107051	
2	9.883638061	18.13908002	1.358029664	12.27124146	0.2339323246	0.4117930113	2.029014065	
3	4.218616925	7.612141719	16.27678418	1.111279142	4.203467798	0.3616919716	-0.6111028449	
4	3.353809703	3.591326189	7.936181817	1.71443185	1.301172298	0.03947521829	-0.6113559859	

5 rows, 23 cols 10 / page Page 1 of 1

```
x,y = df_scaled.drop('Profit',axis=1),df_scaled['Profit']
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2,random_state=42)
```

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error

# Models list
models = [RandomForestRegressor(), SVR(), XGBRegressor()]


# Function for training and evaluating models
def models_training_evaluating(models):
    plt.figure(figsize=(18,5))
    for i, model in enumerate(models):

        # Train the model
        model.fit(x_train, y_train)

        # Make predictions
        y_pred = model.predict(x_test)

        # Calculate Mean Squared Error
        mse = mean_squared_error(y_test, y_pred)

        # Print the result
        print(f"The MSE of {model.__class__.__name__}: {mse} ")
        print('_'*80)

        plt.subplot(1,3,i+1)
        ax = sns.distplot(x=y_test,label='actual data',color='green',kde=True,hist=False)
        sns.distplot(x=y_pred,label='predicted data',color='red',hist=False)
        plt.title(f'the perfomance of {model.__class__.__name__}',size=14)
        plt.legend()
plt.show()


models_training_evaluating(models)
```

```
The MSE of RandomForestRegressor: 0.024898497550406556
________________________________________________________________________________
The MSE of SVR: 0.1311507453425006
________________________________________________________________________________
The MSE of XGBRegressor: 0.13925832976406252
________________________________________________________________________________
```