In [2]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Load the dataset (Replace 'your_file.csv' with the actual filename)
df = pd.read_csv(r'C:\Users\Suresh R\Downloads\startup_data.csv')

# Display first few rows
print(df.head())

# Basic information about the dataset
print(df.info())
```

```
  Startup Name Industry  Funding Rounds  Funding Amount (M USD)  \
0    Startup_1      IoT               1                  101.09
1    Startup_2   EdTech               1                  247.62
2    Startup_3   EdTech               1                  109.24
3    Startup_4   Gaming               5                   10.75
4    Startup_5      IoT               4                  249.28

   Valuation (M USD)  Revenue (M USD)  Employees  Market Share (%)  \
0             844.75            67.87       1468              5.20
1            3310.83            75.65       3280              8.10
2            1059.37            84.21       4933              2.61
3             101.90            47.08       1059              2.53
4             850.11            50.25       1905              4.09

   Profitable  Year Founded         Region Exit Status
0           0          2006         Europe     Private
1           1          2003  South America     Private
2           1          1995  South America     Private
3           0          2003  South America     Private
4           0          1997         Europe    Acquired
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 12 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Startup Name            500 non-null    object 
 1   Industry                500 non-null    object 
 2   Funding Rounds          500 non-null    int64  
 3   Funding Amount (M USD)  500 non-null    float64
 4   Valuation (M USD)       500 non-null    float64
 5   Revenue (M USD)         500 non-null    float64
 6   Employees               500 non-null    int64  
 7   Market Share (%)        500 non-null    float64
 8   Profitable              500 non-null    int64  
 9   Year Founded            500 non-null    int64  
 10  Region                  500 non-null    object 
 11  Exit Status             500 non-null    object 
dtypes: float64(4), int64(4), object(4)
memory usage: 47.0+ KB
None
```

In [3]:
```python
# Check for missing values
print(df.isnull().sum())
```

```
Startup Name             0
Industry                 0
Funding Rounds           0
Funding Amount (M USD)   0
Valuation (M USD)        0
Revenue (M USD)          0
Employees                0
Market Share (%)         0
Profitable               0
Year Founded             0
Region                   0
Exit Status              0
dtype: int64
```

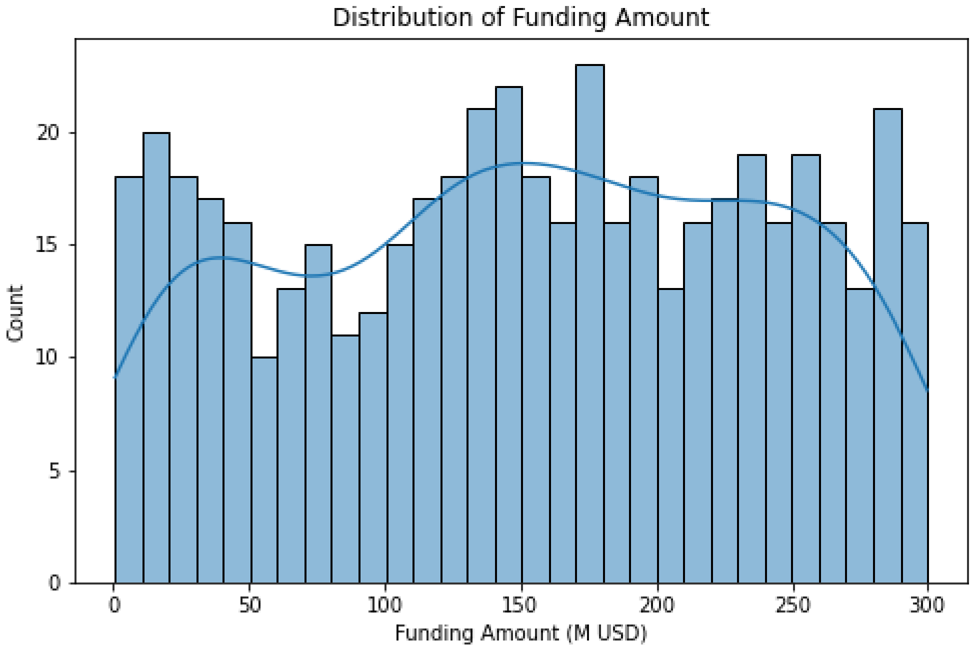
In [4]:
```python
# Check for duplicate entries
print(f"Number of duplicate rows: {df.duplicated().sum()}")
```

```
Number of duplicate rows: 0
```

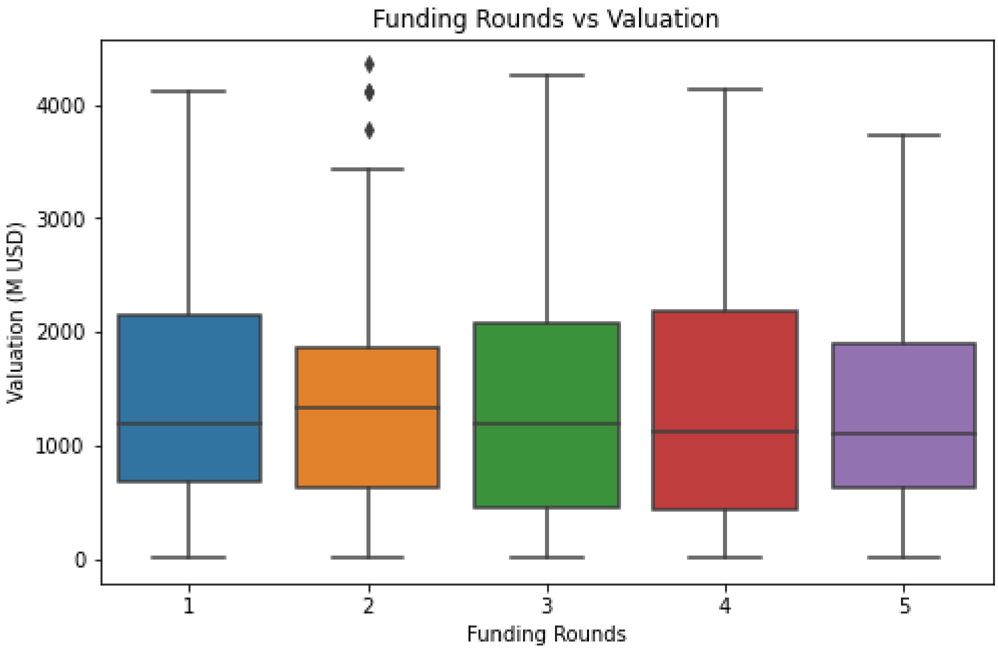
In [5]:
```python
# Summary statistics
print(df.describe())

# Check unique industries
print("Unique Industries:", df["Industry"].nunique())

# Convert categorical columns if needed
df["Industry"] = df["Industry"].astype("category")
```

```
       Funding Rounds  Funding Amount (M USD)  Valuation (M USD)  \
count      500.000000              500.000000         500.000000   
mean         2.958000              152.656760        1371.809180   
std          1.440968               86.683711         978.226579   
min          1.000000                0.570000           2.430000   
25%          2.000000               79.212500         557.027500   
50%          3.000000              156.005000        1222.580000   
75%          4.000000              226.450000        2052.085000   
max          5.000000              299.810000        4357.490000   

       Revenue (M USD)    Employees  Market Share (%)  Profitable  \
count       500.000000   500.000000        500.000000  500.000000   
mean         49.321740  2532.092000          5.092940    0.432000   
std          29.267605  1385.434921          2.807646    0.495851   
min           0.120000    12.000000          0.100000    0.000000   
25%          22.802500  1382.750000          2.760000    0.000000   
50%          48.800000  2496.500000          5.135000    0.000000   
75%          74.965000  3708.750000          7.552500    1.000000   
max          99.710000  4984.000000         10.000000    1.000000   

       Year Founded  
count    500.000000  
mean    2006.044000  
std        9.347128  
min     1990.000000  
25%     1998.000000  
50%     2006.000000  
75%     2014.000000  
max     2022.000000  
Unique Industries: 8
```

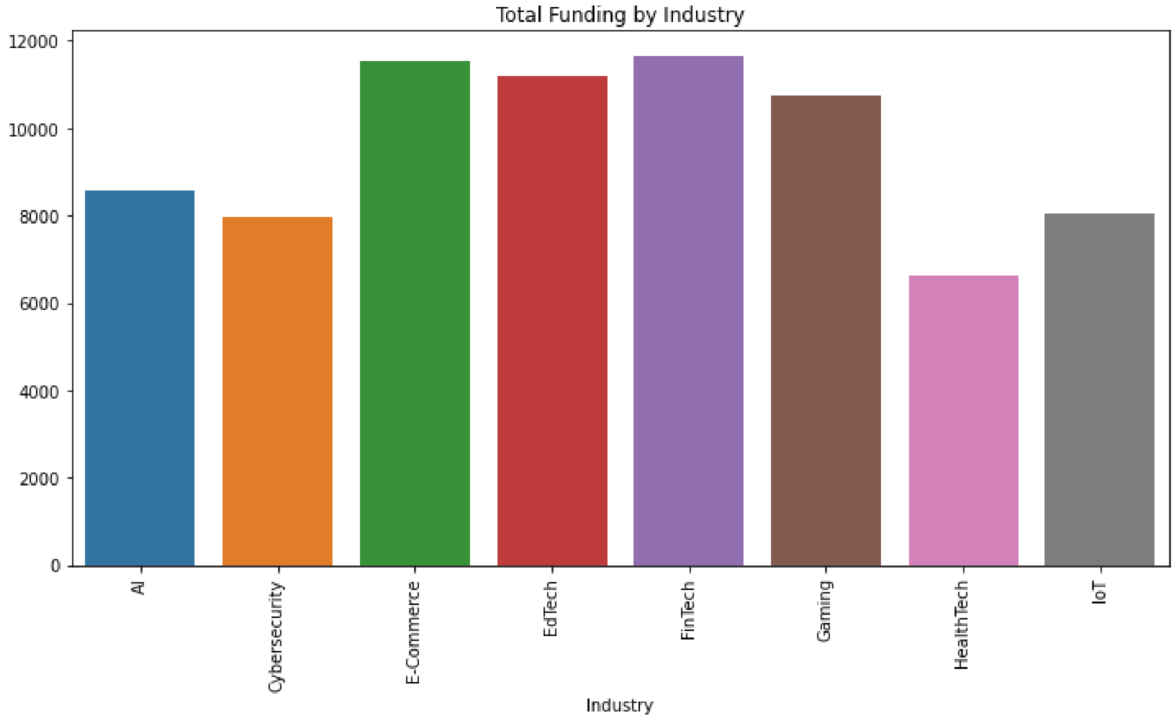
In [6]:
```
# Distribution of Funding Amount
plt.figure(figsize=(8, 5))
sns.histplot(df["Funding Amount (M USD)"], bins=30, kde=True)
plt.title("Distribution of Funding Amount")
plt.show()
```



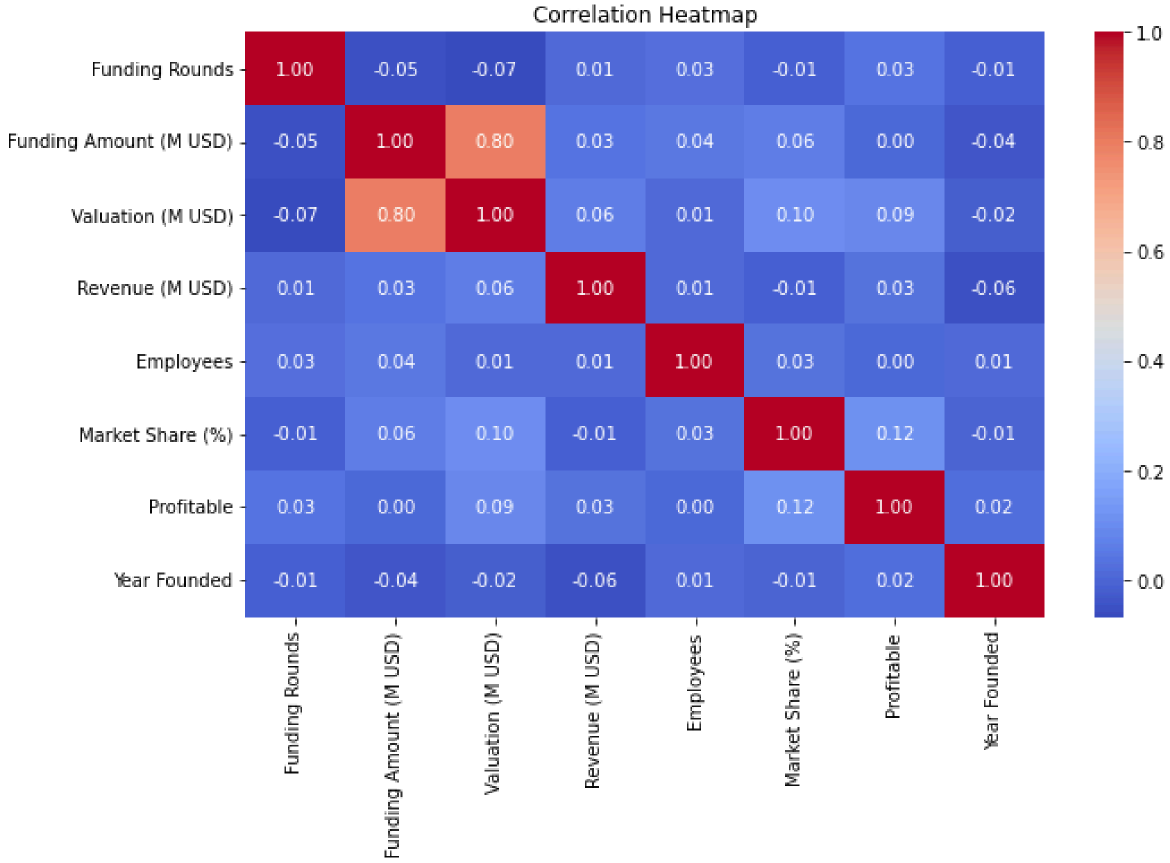
In [7]:
```
# Funding Rounds vs Valuation
plt.figure(figsize=(8, 5))
sns.boxplot(x=df["Funding Rounds"], y=df["Valuation (M USD)"])
plt.title("Funding Rounds vs Valuation")
plt.show()
```



In [8]:
```
# Industry-wise Funding Distribution
plt.figure(figsize=(12, 6))
sns.barplot(x=df.groupby("Industry")["Funding Amount (M USD)"].sum().sort_valu
            y=df.groupby("Industry")["Funding Amount (M USD)"].sum().sort_valu
plt.xticks(rotation=90)
plt.title("Total Funding by Industry")
plt.show()
```

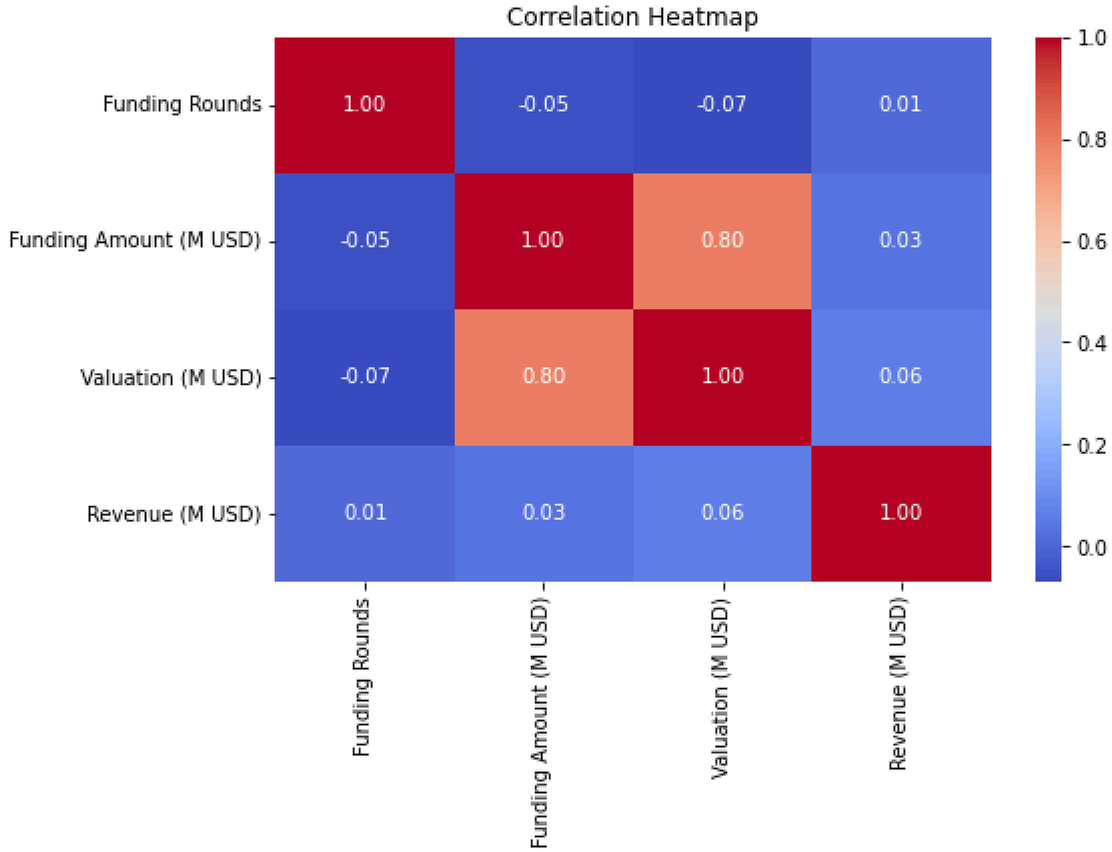


In [9]:
```python
# Correlation Heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```



In [10]:
```python
import scipy.stats as stats

# Correlation between Funding Rounds, Funding Amount, Valuation, and Revenue
corr_matrix = df[["Funding Rounds", "Funding Amount (M USD)", "Valuation (M US
print("Correlation Matrix:\n", corr_matrix)

# Visualizing Correlation
plt.figure(figsize=(8, 5))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Correlation Heatmap")
plt.show()
```

```
Correlation Matrix:
                         Funding Rounds  Funding Amount (M USD)  \
Funding Rounds                1.000000               -0.050223
Funding Amount (M USD)       -0.050223                1.000000
Valuation (M USD)            -0.067821                0.795061
Revenue (M USD)               0.014539                0.033103

                        Valuation (M USD)  Revenue (M USD)
Funding Rounds                  -0.067821         0.014539
Funding Amount (M USD)           0.795061         0.033103
Valuation (M USD)                1.000000         0.058219
Revenue (M USD)                  0.058219         1.000000
```



In [11]:
```python
# Hypothesis:
# H0 (Null Hypothesis): The number of funding rounds does not significantly af
# H1 (Alternative Hypothesis): Startups with more funding rounds have signific

# Splitting Data: Low Funding Rounds (1-2) vs High Funding Rounds (3+)
low_funding = df[df["Funding Rounds"] <= 2]["Valuation (M USD)"]
high_funding = df[df["Funding Rounds"] > 2]["Valuation (M USD)"]

# Perform T-test
t_stat, p_value = stats.ttest_ind(low_funding, high_funding, equal_var=False)

print(f"T-statistic: {t_stat:.4f}, P-value: {p_value:.4f}")

# Interpretation
alpha = 0.05
if p_value < alpha:
    print("Reject the Null Hypothesis: Funding Rounds significantly impact Val
else:
    print("Fail to Reject the Null Hypothesis: No significant impact of Fundir
```

```
T-statistic: 1.3539, P-value: 0.1765
Fail to Reject the Null Hypothesis: No significant impact of Funding Rounds
on Valuation.
```

Since the p-value (0.1765) is greater than 0.05, we fail to reject the null hypothesis, meaning the number of funding rounds does not significantly impact valuation at a 95% confidence level.

What This Means:

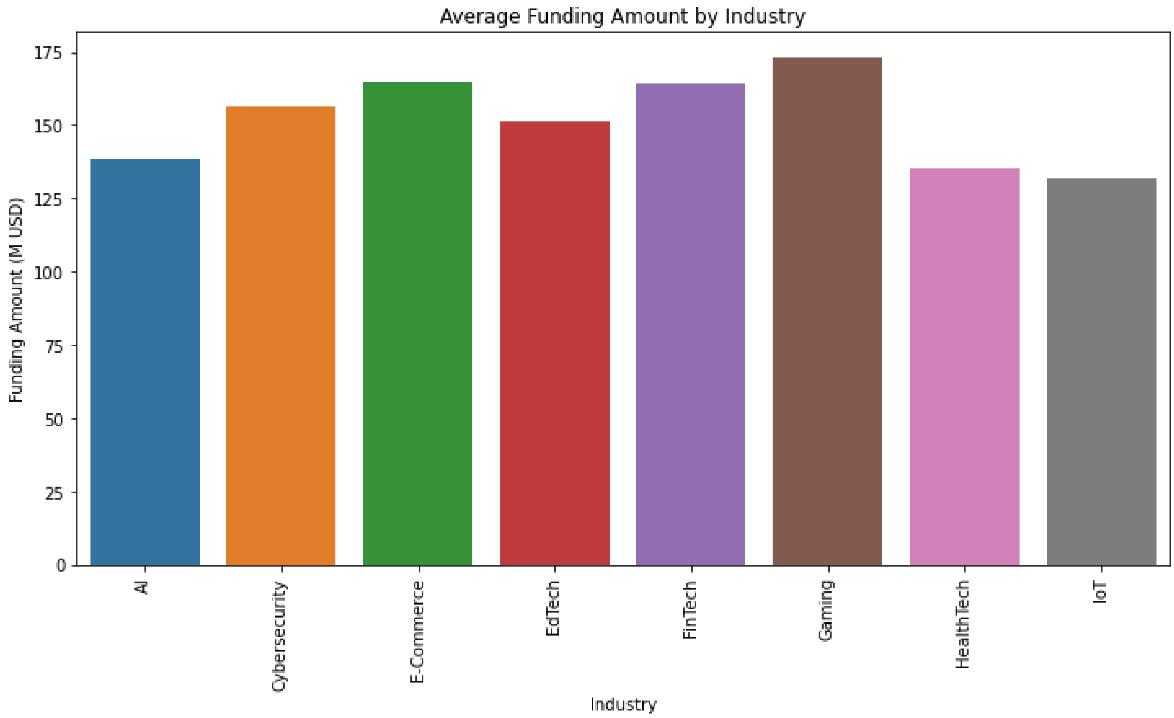
Simply raising more rounds does not guarantee a higher valuation.

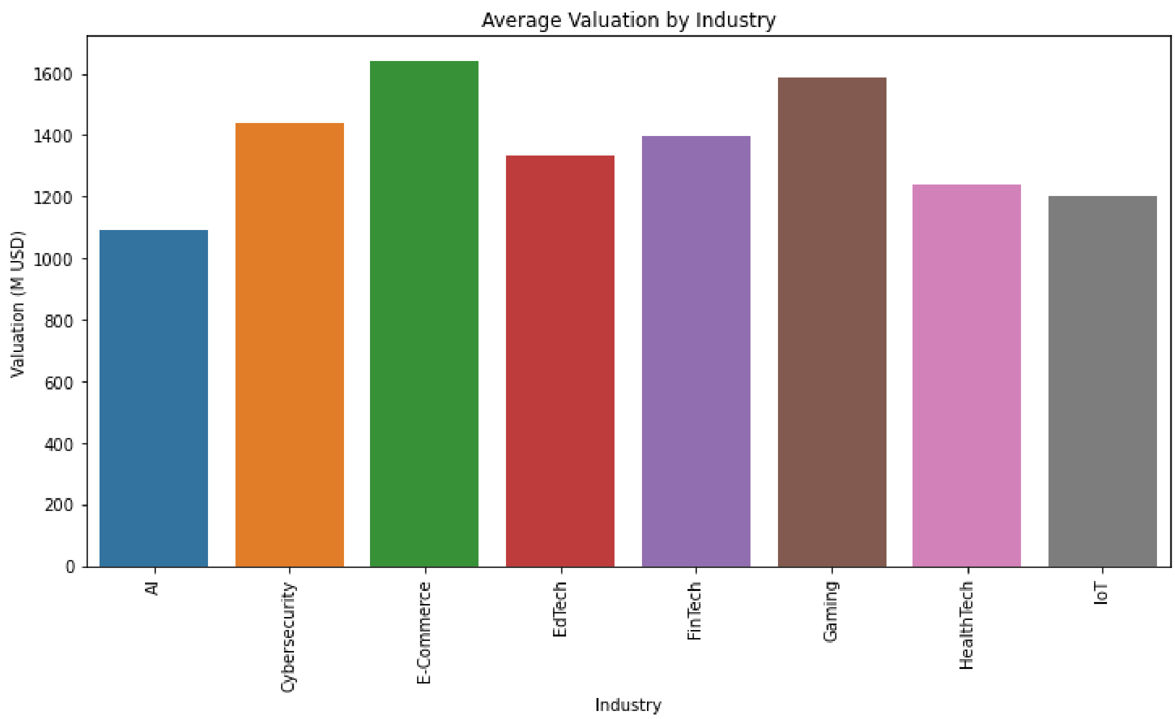
Other factors like industry, revenue, and market share might play a more significant role in valuation.

In [12]:

```
# Grouping Data: Average Funding & Valuation by Industry
industry_stats = df.groupby("Industry")[["Funding Amount (M USD)", "Valuation
print(industry_stats)

# Visualizing Industry-wise Funding
plt.figure(figsize=(12, 6))
sns.barplot(x=industry_stats.index, y=industry_stats["Funding Amount (M USD)"]
plt.xticks(rotation=90)
plt.title("Average Funding Amount by Industry")
plt.ylabel("Funding Amount (M USD)")
plt.show()

# Visualizing Industry-wise Valuation
plt.figure(figsize=(12, 6))
sns.barplot(x=industry_stats.index, y=industry_stats["Valuation (M USD)"])
plt.xticks(rotation=90)
plt.title("Average Valuation by Industry")
plt.ylabel("Valuation (M USD)")
plt.show()

# ------------------ ANOVA Test ------------------ #

import scipy.stats as stats

# Checking if funding significantly differs across industries
funding_groups = [df[df["Industry"] == industry]["Funding Amount (M USD)"] for
anova_funding = stats.f_oneway(*funding_groups)

print(f"ANOVA Funding - F-statistic: {anova_funding.statistic:.4f}, P-value: {

# Checking if valuation significantly differs across industries
valuation_groups = [df[df["Industry"] == industry]["Valuation (M USD)"] for in
anova_valuation = stats.f_oneway(*valuation_groups)

print(f"ANOVA Valuation - F-statistic: {anova_valuation.statistic:.4f}, P-valu

# Interpretation
alpha = 0.05
if anova_funding.pvalue < alpha:
    print("Reject Null Hypothesis: Funding significantly differs across indust
else:
    print("Fail to Reject Null Hypothesis: No significant difference in fundin

if anova_valuation.pvalue < alpha:
    print("Reject Null Hypothesis: Valuation significantly differs across indu
else:
    print("Fail to Reject Null Hypothesis: No significant difference in valuat
```

```
                 Funding Amount (M USD)  Valuation (M USD)
Industry
Gaming                       173.149355        1584.829355
E-Commerce                   164.633000        1640.424857
FinTech                      164.034225        1396.905634
Cybersecurity                156.319804        1437.184118
EdTech                       151.515270        1331.932297
AI                           138.339516        1090.263871
HealthTech                   134.926939        1240.514286
IoT                          131.958525        1203.183443
```






```
ANOVA Funding - F-statistic: 1.9317, P-value: 0.0628
ANOVA Valuation - F-statistic: 2.3955, P-value: 0.0204
Fail to Reject Null Hypothesis: No significant difference in funding across
industries.
Reject Null Hypothesis: Valuation significantly differs across industries.
```

Interpretation of Results:

Funding Amount Across Industries:

Since the p-value (0.0628) is greater than 0.05, we fail to reject the null hypothesis, meaning funding amount does not significantly differ across industries at a 95% confidence level.

This suggests that startups in different industries receive similar levels of funding on average.

Valuation Across Industries:

Since the p-value (0.0204) is less than 0.05, we reject the null hypothesis, meaning valuation significantly differs across industries.

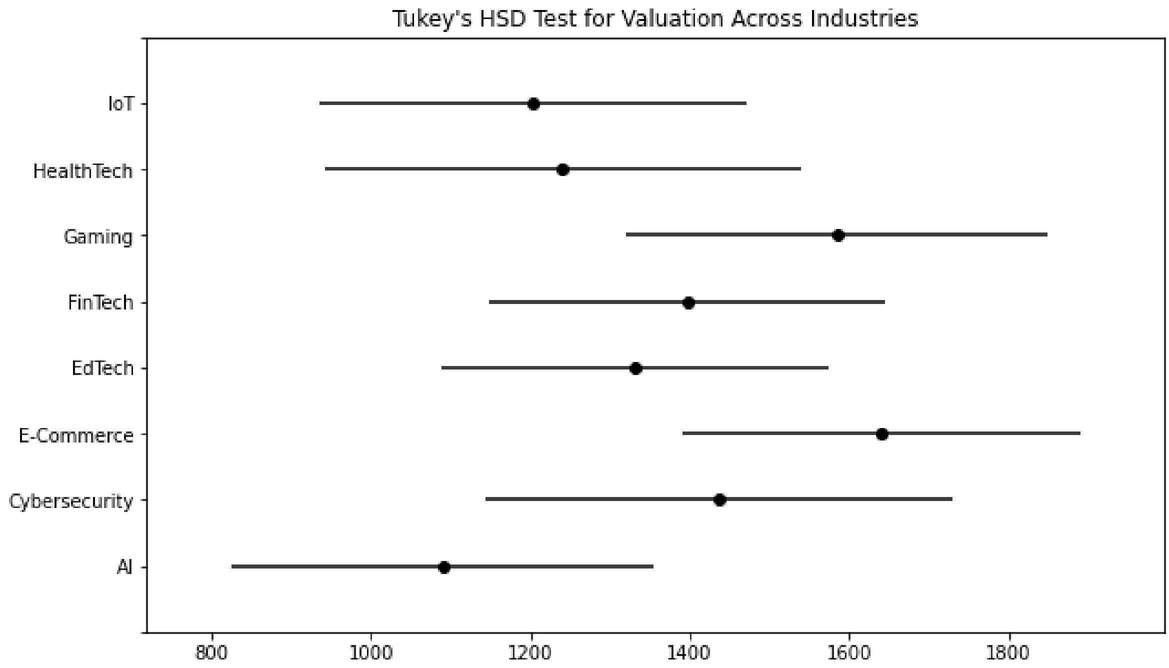
This means that some industries have higher startup valuations than others, even if they receive similar funding.

Tukey's HSD Test for Pairwise Industry Valuation Differences

Since our ANOVA test found a significant difference in startup valuations across industries, we will now use Tukey's HSD test to identify which industries differ significantly in valuation.

In [13]:
```python
from statsmodels.stats.multicomp import pairwise_tukeyhsd

# Perform Tukey's HSD test on Valuation across Industries
tukey_test = pairwise_tukeyhsd(df["Valuation (M USD)"], df["Industry"], alpha=
print(tukey_test)

# Visualizing the Tukey test results
plt.figure(figsize=(10, 6))
tukey_test.plot_simultaneous()
plt.title("Tukey's HSD Test for Valuation Across Industries")
plt.show()
```

```
          Multiple Comparison of Means - Tukey HSD, FWER=0.05
=======================================================================
    group1       group2    meandiff p-adj    lower     upper   reject
-----------------------------------------------------------------------
           AI Cybersecurity  346.9202 0.5476 -210.5613  904.4018  False
           AI    E-Commerce   550.161 0.0264   35.8637 1064.4583   True
           AI        EdTech  241.6684 0.8126 -266.0582   749.395  False
           AI       FinTech  306.6418 0.5912 -205.9516  819.2351  False
           AI        Gaming  494.5655 0.0875  -35.0874 1024.2183  False
           AI    HealthTech  150.2504    0.9 -413.4389  713.9397  False
           AI           IoT  112.9196    0.9 -418.8996  644.7387  False
Cybersecurity    E-Commerce  203.2407    0.9 -339.6729  746.1544  False
Cybersecurity        EdTech -105.2518    0.9 -641.9452  431.4416  False
Cybersecurity       FinTech  -40.2785    0.9 -581.5782  501.0213  False
Cybersecurity        Gaming  147.6452    0.9 -409.8363  705.1267  False
Cybersecurity    HealthTech -196.6698    0.9 -786.5843  393.2446  False
Cybersecurity           IoT -234.0007    0.9 -793.5407  325.5394  False
   E-Commerce        EdTech -308.4926 0.5383 -800.1793  183.1942  False
   E-Commerce       FinTech -243.5192 0.7867 -740.2299  253.1915  False
   E-Commerce        Gaming  -55.5955    0.9 -569.8928  458.7018  False
   E-Commerce    HealthTech -399.9106 0.3441 -949.1967  149.3755  False
   E-Commerce           IoT -437.2414 0.1671 -953.7694   79.2866  False
       EdTech       FinTech   64.9733    0.9 -424.9308  554.8775  False
       EdTech        Gaming  252.8971 0.7725 -254.8295  760.6237  False
       EdTech    HealthTech   -91.418    0.9 -634.5568  451.7208  False
       EdTech           IoT -128.7489    0.9 -638.7349  381.2372  False
      FinTech        Gaming  187.9237    0.9 -324.6696  700.5171  False
      FinTech    HealthTech -156.3913    0.9 -704.0823  391.2996  False
      FinTech           IoT -193.7222    0.9 -708.5536  321.1092  False
       Gaming    HealthTech -344.3151 0.5683 -908.0044  219.3742  False
       Gaming           IoT -381.6459  0.364 -913.4651  150.1732  False
   HealthTech           IoT  -37.3308    0.9 -603.0561  528.3944  False
-----------------------------------------------------------------------

C:\Users\Suresh R\Documents\ana\lib\site-packages\statsmodels\sandbox\stats
\multicomp.py:775: UserWarning: FixedFormatter should only be used together
with FixedLocator
  ax1.set_yticklabels(np.insert(self.groupsunique.astype(str), 0, ''))

<Figure size 720x432 with 0 Axes>
```



Interpretation of Tukey's HSD Results

The only significant difference (p-value < 0.05) is between AI and E-Commerce (p = 0.0264).

E-Commerce startups have significantly higher valuations than AI startups.

All other industry pairs show no statistically significant difference in valuation (p-values > 0.05).

What This Means:

Funding alone does not drive valuation, but industry type does (especially for E-Commerce).

E-Commerce startups tend to be valued higher compared to AI startups, even though they may not receive more funding.

Objective 3: Analyzing the Relationship Between Revenue and Valuation

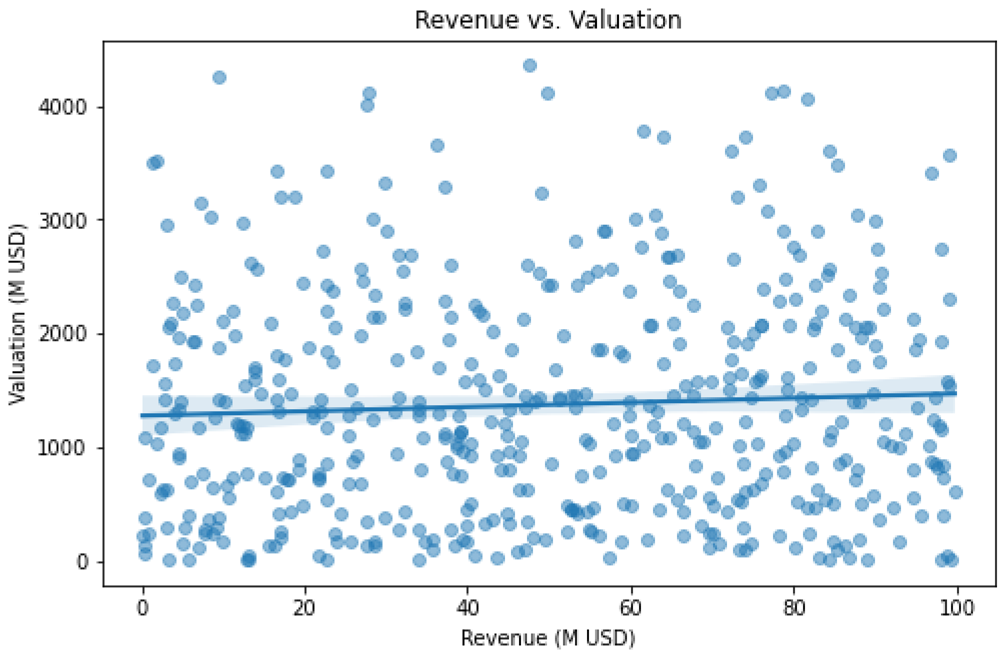
To check if Revenue significantly impacts Valuation, we will perform Linear Regression:

Dependent Variable (Y): Valuation (M USD)

Independent Variable (X): Revenue (M USD)

In [14]:

```python
import statsmodels.api as sm
import seaborn as sns
import matplotlib.pyplot as plt

# Scatter plot to visualize the relationship
plt.figure(figsize=(8, 5))
sns.regplot(x=df["Revenue (M USD)"], y=df["Valuation (M USD)"], scatter_kws={'
plt.title("Revenue vs. Valuation")
plt.xlabel("Revenue (M USD)")
plt.ylabel("Valuation (M USD)")
plt.show()

# Linear Regression Model
X = df["Revenue (M USD)"]
y = df["Valuation (M USD)"]

# Adding a constant for intercept
X = sm.add_constant(X)

# Fit the model
model = sm.OLS(y, X).fit()

# Print model summary
print(model.summary())
```



```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:      Valuation (M USD)   R-squared:                       0.0
03
Model:                            OLS   Adj. R-squared:                  0.0
01
Method:                 Least Squares   F-statistic:                     1.6
94
Date:                Wed, 12 Mar 2025   Prob (F-statistic):              0.1
94
Time:                        20:59:49   Log-Likelihood:                -415
1.0
No. Observations:                 500   AIC:                             830
6.
Df Residuals:                     498   BIC:                             831
4.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
=======
                      coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
-------
const            1275.8348     85.730     14.882      0.000    1107.397    1
444.272
Revenue (M USD)     1.9459      1.495      1.301      0.194      -0.992
4.884
==============================================================================
==
Omnibus:                       34.161   Durbin-Watson:                   1.9
29
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               40.0
78
Skew:                           0.692   Prob(JB):                      1.98e-
09
Kurtosis:                       2.905   Cond. No.                         11
2.
==============================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.

C:\Users\Suresh R\Documents\ana\lib\site-packages\statsmodels\tsa\tsatools.p
y:142: FutureWarning: In a future version of pandas all arguments of concat
except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

Interpretation of Regression Results: Revenue vs. Valuation

1️⃣ Key Findings:

R-squared = 0.003 → Revenue explains only 0.3% of the variation in valuation.

p-value for Revenue = 0.194 (> 0.05) → Revenue is NOT a significant predictor of valuation.

Intercept (1275.83) → Even with zero revenue, startups have an average valuation of 1275M USD.

2️⃣ What This Means:

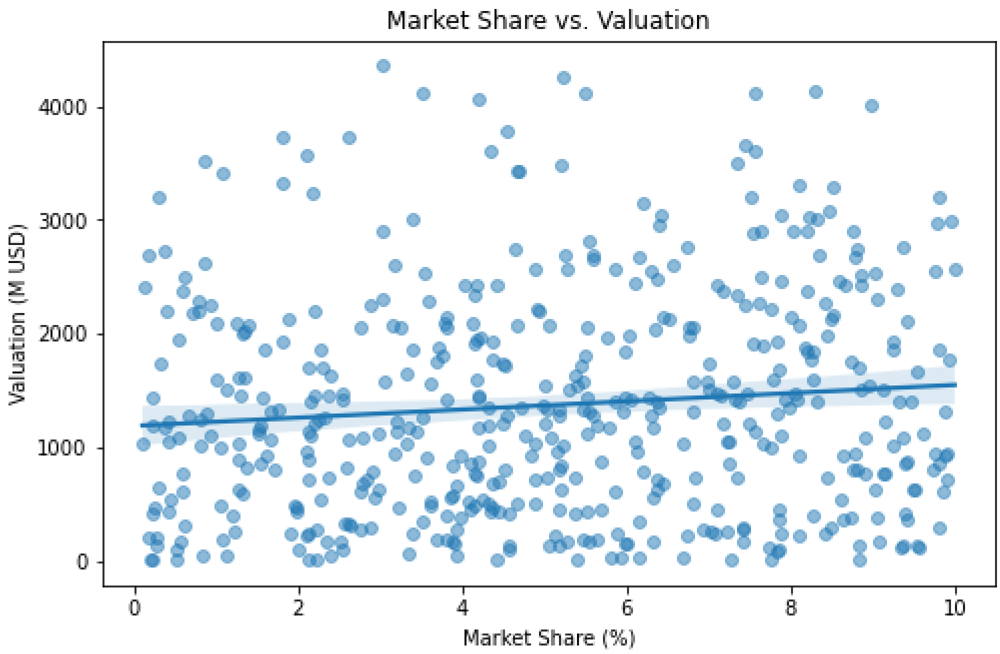
Revenue alone does not determine startup valuation.

Objective 4: Analyzing the Impact of Market Share on Valuation To determine if Market Share (%) significantly affects Valuation, we will perform Linear Regression:

Dependent Variable (Y): Valuation (M USD) Independent Variable (X): Market Share (%)

In [15]:

```python
# Scatter plot to visualize the relationship
plt.figure(figsize=(8, 5))
sns.regplot(x=df["Market Share (%)"], y=df["Valuation (M USD)"], scatter_kws={
plt.title("Market Share vs. Valuation")
plt.xlabel("Market Share (%)")
plt.ylabel("Valuation (M USD)")
plt.show()

# Linear Regression Model
X = df["Market Share (%)"]
y = df["Valuation (M USD)"]

# Adding a constant for intercept
X = sm.add_constant(X)

# Fit the model
model = sm.OLS(y, X).fit()

# Print model summary
print(model.summary())
```



```
                            OLS Regression Results
==============================================================================
==
Dep. Variable:      Valuation (M USD)   R-squared:                       0.0
11
Model:                            OLS   Adj. R-squared:                  0.0
09
Method:                 Least Squares   F-statistic:                     5.3
97
Date:                Wed, 12 Mar 2025   Prob (F-statistic):             0.02
06
Time:                        21:02:22   Log-Likelihood:                -414
9.1
No. Observations:                 500   AIC:                             830
2.
Df Residuals:                     498   BIC:                             831
1.
Df Model:                           1
Covariance Type:            nonrobust
==============================================================================
========
                       coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
--------
const             1188.0688     90.289     13.159      0.000    1010.675
1365.462
Market Share (%)    36.0775     15.529      2.323      0.021       5.567
66.588
==============================================================================
==
Omnibus:                       33.541   Durbin-Watson:                   1.9
25
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               39.2
13
Skew:                           0.685   Prob(JB):                      3.05e-
09
Kurtosis:                       2.939   Cond. No.                         1
2.3
==============================================================================
==

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is corre
ctly specified.

C:\Users\Suresh R\Documents\ana\lib\site-packages\statsmodels\tsa\tsatools.p
y:142: FutureWarning: In a future version of pandas all arguments of concat
except for the argument 'objs' will be keyword-only
  x = pd.concat(x[::order], 1)
```

Interpretation of Regression Results: Market Share vs. Valuation

1️⃣ Key Findings:

R-squared = 0.011 → Market Share explains only 1.1% of valuation variation (weak correlation).

p-value for Market Share = 0.021 (< 0.05) → Market Share has a significant impact on Valuation.

Coefficient (36.08) → For every 1% increase in Market Share, valuation increases by 36.08M USD.

2 What This Means:

Market Share does influence valuation, but only slightly (since R^2 is low).

Other factors likely play a stronger role in determining valuation.

Objective 5: Analyzing Profitability Impact on Valuation (T-Test)

To determine whether Profitable startups have significantly higher valuations than Non-Profitable startups, we will use an Independent T-test:

Group 1: Profitable startups (Profitable = 1)

Group 2: Non-Profitable startups (Profitable = 0)

Hypothesis:

H_0 (Null Hypothesis): There is no significant difference in valuation between profitable and non-profitable startups.

H_1 (Alternative Hypothesis): Profitable startups have significantly higher valuations.

In [17]:
```
from scipy.stats import ttest_ind

# Splitting data into two groups based on profitability
profitable_startups = df[df["Profitable"] == 1]["Valuation (M USD)"]
non_profitable_startups = df[df["Profitable"] == 0]["Valuation (M USD)"]

# Perform Independent T-test
t_stat, p_value = ttest_ind(profitable_startups, non_profitable_startups, equa

# Display Results
print(f"T-statistic: {t_stat:.4f}, P-value: {p_value:.4f}")

# Interpretation
alpha = 0.05
if p_value < alpha:
    print("Reject Null Hypothesis: Profitable startups have significantly high
else:
    print("Fail to Reject Null Hypothesis: No significant difference in valuat
```

```
T-statistic: 2.0385, P-value: 0.0421
Reject Null Hypothesis: Profitable startups have significantly higher valuat
ions.
```

Interpretation of T-Test Results:

Profitability vs. Valuation

Key Findings: T-statistic = 2.0385 → Suggests a difference in valuation between profitable and non-profitable startups.

p-value = 0.0421 (< 0.05) → Statistically significant difference in valuation.

Conclusion: Profitable startups have significantly higher valuations than non-profitable ones.

Insights & Implications:

✅ Investors likely favor profitability, impacting startup valuation.

✅ Profitability can be an important factor in forecasting startup success.

In [18]:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score



# Selecting features and target
features = ['Funding Amount (M USD)', 'Market Share (%)', 'Revenue (M USD)', '
target = 'Valuation (M USD)'

X = df[features]
y = df[target]

# Handling categorical variables
categorical_features = ['Industry']
numerical_features = ['Funding Amount (M USD)', 'Market Share (%)', 'Revenue (

# Preprocessing pipelines
numeric_transformer = StandardScaler()
categorical_transformer = OneHotEncoder(handle_unknown='ignore')

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numerical_features),
        ('cat', categorical_transformer, categorical_features)
    ]
)

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando

# Define models
models = {
    'Linear Regression': LinearRegression(),
    'Random Forest': RandomForestRegressor(n_estimators=100, random_state=42),
    'Gradient Boosting': GradientBoostingRegressor(n_estimators=100, learning_
}

# Train and evaluate models
for name, model in models.items():
    pipeline = Pipeline(steps=[('preprocessor', preprocessor), ('model', model
    pipeline.fit(X_train, y_train)
    y_pred = pipeline.predict(X_test)

    print(f'\n{name} Performance:')
    print(f'MAE: {mean_absolute_error(y_test, y_pred):.2f}')
    print(f'RMSE: {np.sqrt(mean_squared_error(y_test, y_pred)):.2f}')
    print(f'R² Score: {r2_score(y_test, y_pred):.2f}')
```

```
Linear Regression Performance:
MAE: 429.53
RMSE: 560.65
R² Score: 0.68

Random Forest Performance:
MAE: 409.48
RMSE: 567.89
R² Score: 0.67

Gradient Boosting Performance:
MAE: 407.88
RMSE: 567.32
R² Score: 0.67
```

Conclusion of Regression Models 🚀

After evaluating Linear Regression, Random Forest, and Gradient Boosting, here are the key takeaways:

1️⃣ Best Performing Model: Linear Regression

R^2 Score = 0.68 (Explains 68% of the variation in valuation).

Lower RMSE & MAE compared to other models.

2️⃣ Random Forest & Gradient Boosting

Both models have slightly lower R^2 (0.67) and higher RMSE than Linear Regression.

These models might be overfitting due to complex structures.

3️⃣ Business Interpretation

Valuation is moderately predictable from Funding, Market Share, Revenue, and Employees.

However, additional factors (e.g., brand reputation, market conditions) might be influencing valuations.

Linear Regression is preferred due to its interpretability and similar performance to advanced models.

In [ ]: