In [122]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
```

In [123]:
```
df = pd.read_csv("C:\\Users\\Dell\\Downloads\\investments_VC.csv", enco
df.head()
```

Out[123]:

	permalink	name	homepage_url	
0	/organization/waywire	#waywire	http://www.waywire.com	|Entertainment
1	/organization/tv-communications	&TV Communications	http://enjoyandtv.com	
2	/organization/rock-your-paper	'Rock' Your Paper	http://www.rockyourpaper.org	
3	/organization/in-touch-network	(In)Touch Network	http://www.InTouchNetwork.com	|Electronics|Guides|(
4	/organization/r-ranch-and-mine	-R- Ranch and Mine	NaN	|Tou

5 rows × 39 columns

In [124]:
```
df.shape
```

Out[124]: (54294, 39)

In [125]:
```
data_cleaned = df.drop_duplicates()
```

In [126]:
```
data_cleaned.to_csv('cleaned_investments_VC.csv', index=False)
```

In [127]:
```
print("Original number of rows:", df.shape[0])
print("Number of rows after removing duplicates:", data_cleaned.shape[0
```

```
Original number of rows: 54294
Number of rows after removing duplicates: 49439
```

In [128]:
```
data_cleaned.shape
```

Out[128]: (49439, 39)

In [129]:
```
data = data_cleaned
data.shape
```

Out[129]: (49439, 39)

In [130]:
```
data.head(5)
```

Out[130]:

	permalink	name	homepage_url	
0	/organization/waywire	#waywire	http://www.waywire.com	|Entertainment
1	/organization/tv-communications	&TV Communications	http://enjoyandtv.com	
2	/organization/rock-your-paper	'Rock' Your Paper	http://www.rockyourpaper.org	
3	/organization/in-touch-network	(In)Touch Network	http://www.InTouchNetwork.com	|Electronics|Guides|(
4	/organization/r-ranch-and-mine	-R- Ranch and Mine	NaN	|Tou

5 rows × 39 columns

In [131]:
```
data[' funding_total_usd '] = data[' funding_total_usd '].replace('[\$,
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_25860\3776153038.py:1: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
  data[' funding_total_usd '] = data[' funding_total_usd '].replace
('[\$,]', '', regex=True)
```

In [132]:
```
data[' funding_total_usd '] = pd.to_numeric(data[' funding_total_usd ']
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_25860\3158862307.py:1: Sett
ingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas
-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
(https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
#returning-a-view-versus-a-copy)
  data[' funding_total_usd '] = pd.to_numeric(data[' funding_total_usd
'], errors='coerce')
```

In [133]:
```
print(data[' funding_total_usd '].dtype)
print(data[' funding_total_usd '].head())
```

```
float64
0    1750000.0
1    4000000.0
2      40000.0
3    1500000.0
4      60000.0
Name:  funding_total_usd , dtype: float64
```

In [134]:
```
# List of columns to drop (e.g., URLs, incomplete columns)
columns_to_drop = ['homepage_url', 'permalink', 'region', 'state_code',

# Drop the columns from the dataset
data_clean = data.drop(columns=columns_to_drop, axis=1)

# Save the cleaned dataset (optional)
data_clean.to_csv('clean_dataset.csv', index=False)

# Show the remaining columns after dropping
print("Remaining columns:", data_clean.columns)
```

```
Remaining columns: Index(['name', 'category_list', ' market ', ' fundi
ng_total_usd ', 'status',
       'country_code', 'funding_rounds', 'founded_at', 'founded_mont
h',
       'founded_quarter', 'founded_year', 'first_funding_at',
       'last_funding_at', 'seed', 'venture', 'equity_crowdfunding',
       'undisclosed', 'convertible_note', 'debt_financing', 'angel',
'grant',
       'private_equity', 'post_ipo_equity', 'post_ipo_debt',
       'secondary_market', 'product_crowdfunding', 'round_A', 'round_
B',
       'round_C', 'round_D', 'round_E', 'round_F', 'round_G', 'round_
H'],
      dtype='object')
```

In [135]:
```
data1 = data_clean
data1.shape
```

Out[135]: (49439, 34)

In [136]:
```
data1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 49439 entries, 0 to 49438
Data columns (total 34 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   name                  49437 non-null  object
 1   category_list         45477 non-null  object
 2    market               45470 non-null  object
 3    funding_total_usd    40907 non-null  float64
 4   status                48124 non-null  object
 5   country_code          44165 non-null  object
 6   funding_rounds        49438 non-null  float64
 7   founded_at            38554 non-null  object
 8   founded_month         38482 non-null  object
 9   founded_quarter       38482 non-null  object
 10  founded_year          38482 non-null  float64
 11  first_funding_at      49438 non-null  object
 12  last_funding_at       49438 non-null  object
 13  seed                  49438 non-null  float64
 14  venture               49438 non-null  float64
 15  equity_crowdfunding   49438 non-null  float64
 16  undisclosed           49438 non-null  float64
 17  convertible_note      49438 non-null  float64
 18  debt_financing        49438 non-null  float64
 19  angel                 49438 non-null  float64
 20  grant                 49438 non-null  float64
 21  private_equity        49438 non-null  float64
 22  post_ipo_equity       49438 non-null  float64
 23  post_ipo_debt         49438 non-null  float64
 24  secondary_market      49438 non-null  float64
 25  product_crowdfunding  49438 non-null  float64
 26  round_A               49438 non-null  float64
 27  round_B               49438 non-null  float64
 28  round_C               49438 non-null  float64
 29  round_D               49438 non-null  float64
 30  round_E               49438 non-null  float64
 31  round_F               49438 non-null  float64
 32  round_G               49438 non-null  float64
 33  round_H               49438 non-null  float64
dtypes: float64(24), object(10)
memory usage: 13.2+ MB
```

In [137]:
```
# Drop rows with any missing values
data_cleaned_rows = data1.dropna()

# Drop columns with missing values
data_cleaned_cols = data1.dropna(axis=1)

# Show the number of rows/columns after dropping
print("Data shape after dropping rows with missing values:", data_clean
print("Data shape after dropping columns with missing values:", data_cl
```

```
Data shape after dropping rows with missing values: (28292, 34)
Data shape after dropping columns with missing values: (49439, 0)
```

In [138]:
```
# Fill missing values with a specific value, such as 0
data_filled = data1.fillna(0, inplace = True)

# Fill missing numerical columns with the mean value
data1['funding_total_usd'] = data1[' funding_total_usd '].fillna(data1[

# Fill missing categorical columns with the most frequent value (mode)
data1['status'] = data1['status'].fillna(data1['status'].mode()[0])

# Check the updated dataset
print(data1.info())
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 49439 entries, 0 to 49438
Data columns (total 35 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   name                  49439 non-null  object
 1   category_list         49439 non-null  object
 2    market               49439 non-null  object
 3    funding_total_usd    49439 non-null  float64
 4   status                49439 non-null  object
 5   country_code          49439 non-null  object
 6   funding_rounds        49439 non-null  float64
 7   founded_at            49439 non-null  object
 8   founded_month         49439 non-null  object
 9   founded_quarter       49439 non-null  object
 10  founded_year          49439 non-null  float64
 11  first_funding_at      49439 non-null  object
 12  last_funding_at       49439 non-null  object
 13  seed                  49439 non-null  float64
 14  venture               49439 non-null  float64
 15  equity_crowdfunding   49439 non-null  float64
 16  undisclosed           49439 non-null  float64
 17  convertible_note      49439 non-null  float64
 18  debt_financing        49439 non-null  float64
 19  angel                 49439 non-null  float64
 20  grant                 49439 non-null  float64
 21  private_equity        49439 non-null  float64
 22  post_ipo_equity       49439 non-null  float64
 23  post_ipo_debt         49439 non-null  float64
 24  secondary_market      49439 non-null  float64
 25  product_crowdfunding  49439 non-null  float64
 26  round_A               49439 non-null  float64
 27  round_B               49439 non-null  float64
 28  round_C               49439 non-null  float64
 29  round_D               49439 non-null  float64
 30  round_E               49439 non-null  float64
 31  round_F               49439 non-null  float64
 32  round_G               49439 non-null  float64
 33  round_H               49439 non-null  float64
 34  funding_total_usd     49439 non-null  float64
dtypes: float64(25), object(10)
memory usage: 13.6+ MB
None
```

In [139]:
```
data1.shape
```

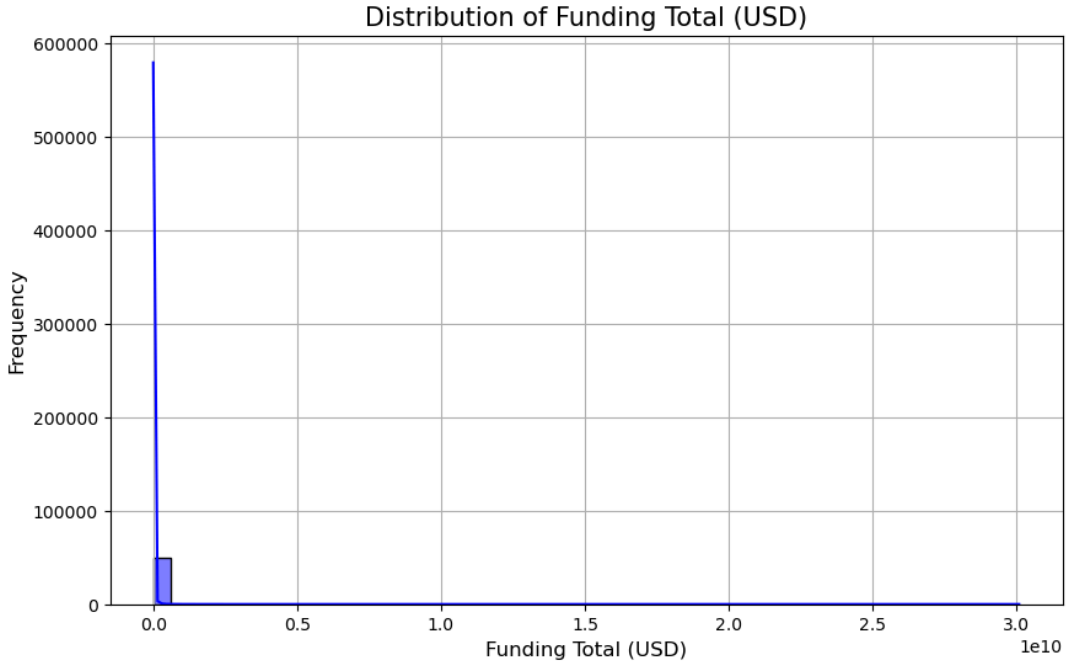
Out[139]: (49439, 35)

In [140]:
```
# 1. Convert date columns (founded_at, first_funding_at) to datetime
data1['founded_at'] = pd.to_datetime(data1['founded_at'], errors='coerc
data1['first_funding_at'] = pd.to_datetime(data1['first_funding_at'], e

# 2. Convert numeric columns (e.g., funding_total_usd) to numeric
data1[' funding_total_usd '] = pd.to_numeric(data1[' funding_total_usd

# 3. Handle categorical columns (converting to category type or using o
# Example: Converting 'status' column to categorical type
data1['status'] = data1['status'].astype('category')

# If needed, you can also apply one-hot encoding for categorical variab
# Example: One-hot encoding the 'status' column for analysis
data1 = pd.get_dummies(data1, columns=['status'], drop_first=True)

# Check the data after transformations
print(data1.info())
print(data1[['founded_at', 'first_funding_at', ' funding_total_usd ']].
```

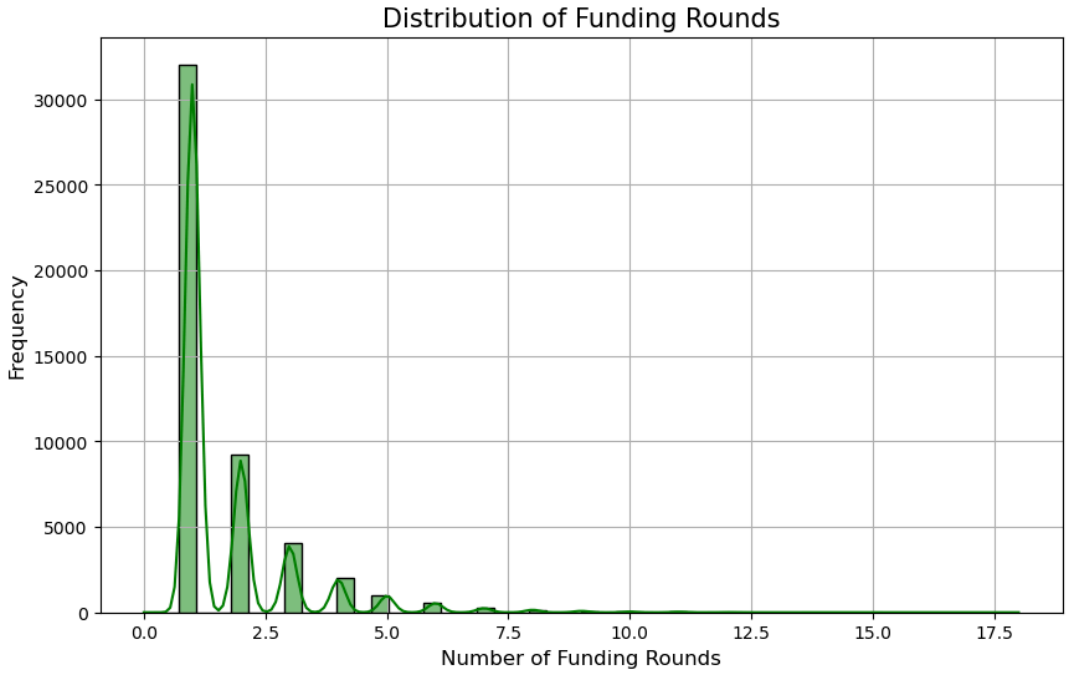
```
<class 'pandas.core.frame.DataFrame'>
Index: 49439 entries, 0 to 49438
Data columns (total 37 columns):
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   name                  49439 non-null  object
 1   category_list         49439 non-null  object
 2    market               49439 non-null  object
 3    funding_total_usd    49439 non-null  float64
 4   country_code          49439 non-null  object
 5   funding_rounds        49439 non-null  float64
 6   founded_at            38553 non-null  datetime64[ns]
 7   founded_month         49439 non-null  object
 8   founded_quarter       49439 non-null  object
 9   founded_year          49439 non-null  float64
 10  first_funding_at      49428 non-null  datetime64[ns]
 11  last_funding_at       49439 non-null  object
 12  seed                  49439 non-null  float64
 13  venture               49439 non-null  float64
```

Understanding Distribution:

In [141]:
```
# 1. Plot distribution for 'funding_total_usd'
plt.figure(figsize=(10, 6))
sns.histplot(data1[' funding_total_usd '], bins=50, kde=True, color='bl
plt.title('Distribution of Funding Total (USD)', fontsize=15)
plt.xlabel('Funding Total (USD)', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.grid(True)
plt.show()
```

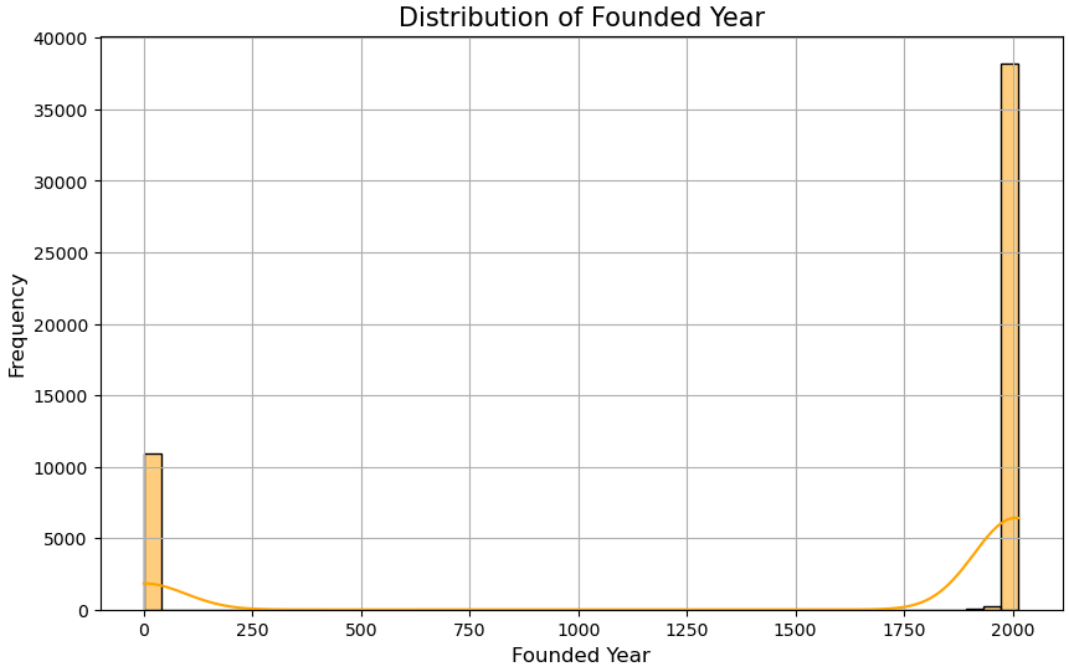


In [142]:
```
# 2. Plot distribution for 'funding_rounds'
plt.figure(figsize=(10, 6))
sns.histplot(data1['funding_rounds'], bins=50, kde=True, color='green')
plt.title('Distribution of Funding Rounds', fontsize=15)
plt.xlabel('Number of Funding Rounds', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.grid(True)
plt.show()
```



In [143]:

```
# 3. Plot distribution for 'founded_year'
plt.figure(figsize=(10, 6))
sns.histplot(data1['founded_year'], bins=50, kde=True, color='orange')
plt.title('Distribution of Founded Year', fontsize=15)
plt.xlabel('Founded Year', fontsize=12)
plt.ylabel('Frequency', fontsize=12)
plt.grid(True)
plt.show()
```



In [144]:
```
# Optionally, print summary statistics to identify outliers or anomalie
print("Summary statistics for Funding Total (USD):\n", data1[' funding_
print("\nSummary statistics for Funding Rounds:\n", data1['funding_roun
print("\nSummary statistics for Founded Year:\n", data1['founded_year']
```

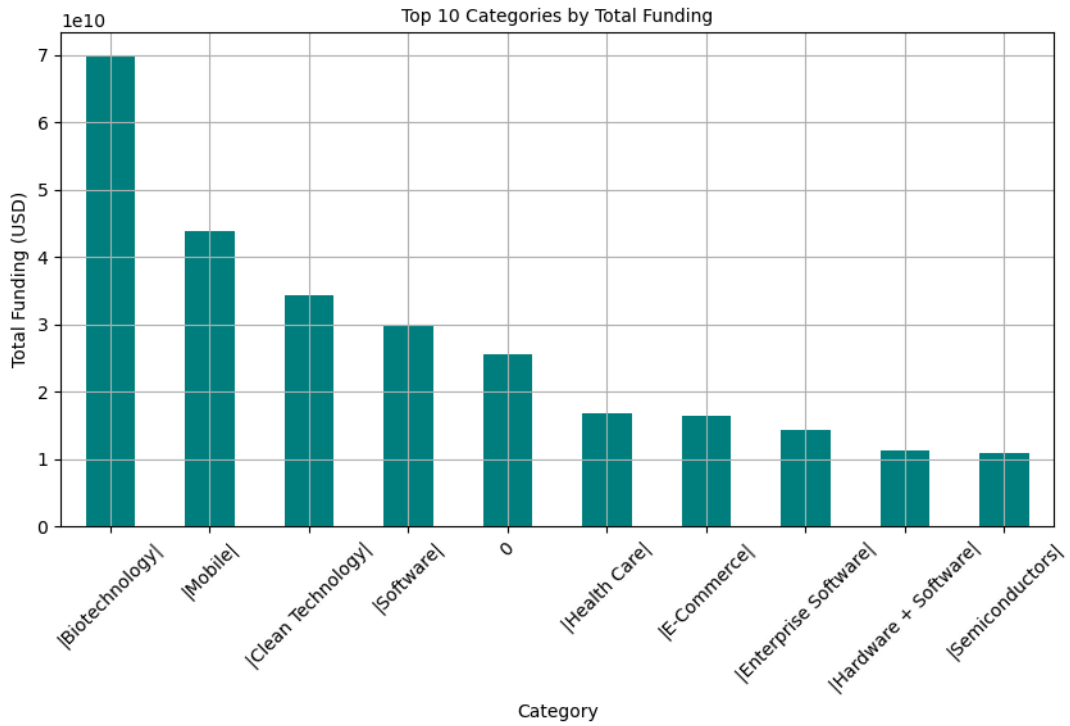
```
Summary statistics for Funding Total (USD):
 count    4.943900e+04
mean     1.316640e+07
std      1.535525e+08
min      0.000000e+00
25%      5.000000e+04
50%      1.000000e+06
75%      6.771937e+06
max      3.007950e+10
Name:  funding_total_usd , dtype: float64

Summary statistics for Funding Rounds:
 count    49439.000000
mean         1.696171
std          1.294222
min          0.000000
25%          1.000000
50%          1.000000
75%          2.000000
max         18.000000
Name: funding_rounds, dtype: float64

Summary statistics for Founded Year:
 count    49439.000000
mean      1562.474848
std        833.774114
min          0.000000
25%       1993.000000
50%       2008.000000
75%       2011.000000
max       2014.000000
Name: founded_year, dtype: float64
```

Catagory & Market Analysis:

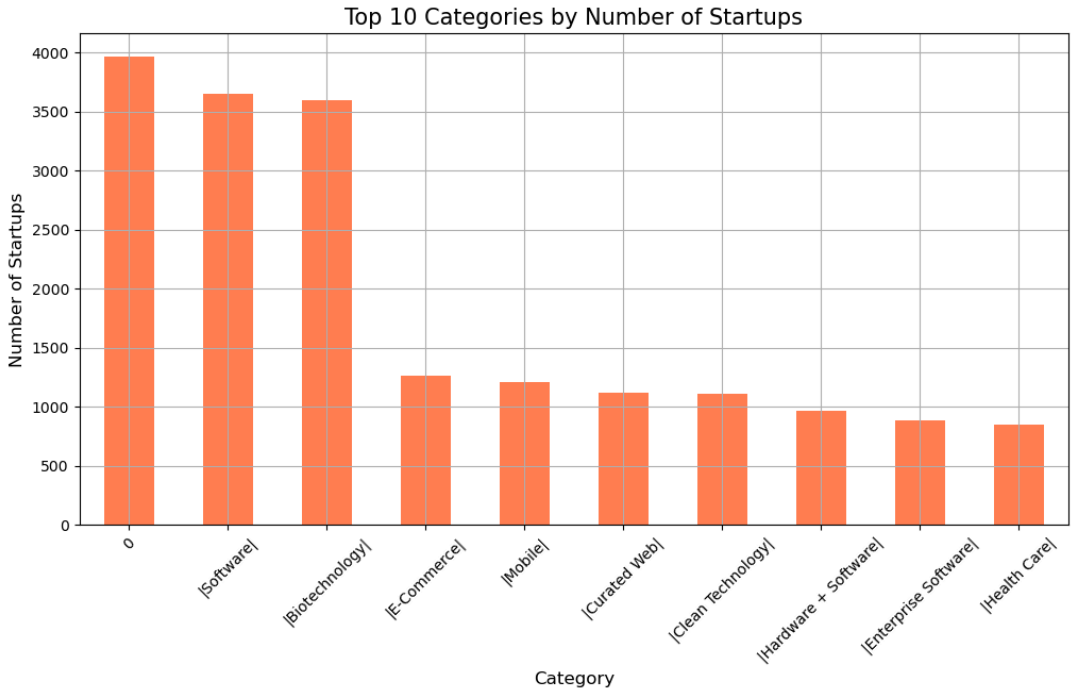
In [145]:
```
# 1. Analyze the distribution of startups by 'category_list'
category_funding = data1.groupby('category_list')[' funding_total_usd '
category_startups = data1['category_list'].value_counts()
```

In [151]:

```
# 2. Visualize the top 10 categories by total funding
plt.figure(figsize=(10, 5))
category_funding.head(10).plot(kind='bar', color='teal')
plt.title('Top 10 Categories by Total Funding', fontsize=10)
plt.xlabel('Category', fontsize=10)
plt.ylabel('Total Funding (USD)', fontsize=10)
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



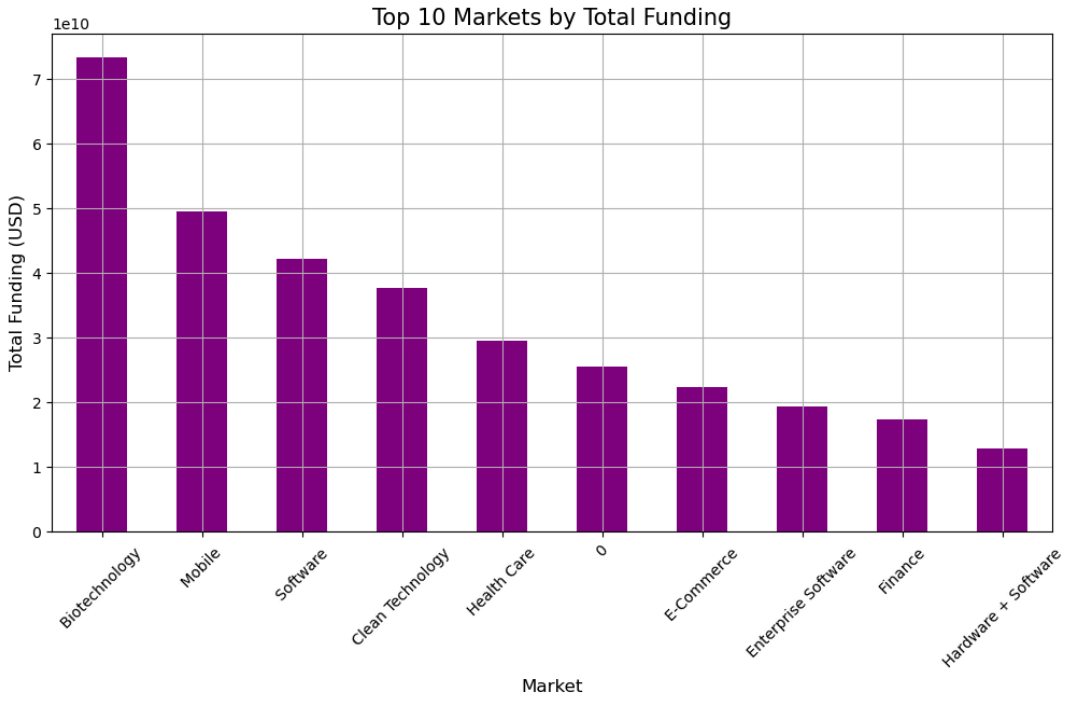
In [152]:
```
# 3. Visualize the top 10 categories by number of startups
plt.figure(figsize=(12, 6))
category_startups.head(10).plot(kind='bar', color='coral')
plt.title('Top 10 Categories by Number of Startups', fontsize=15)
plt.xlabel('Category', fontsize=12)
plt.ylabel('Number of Startups', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



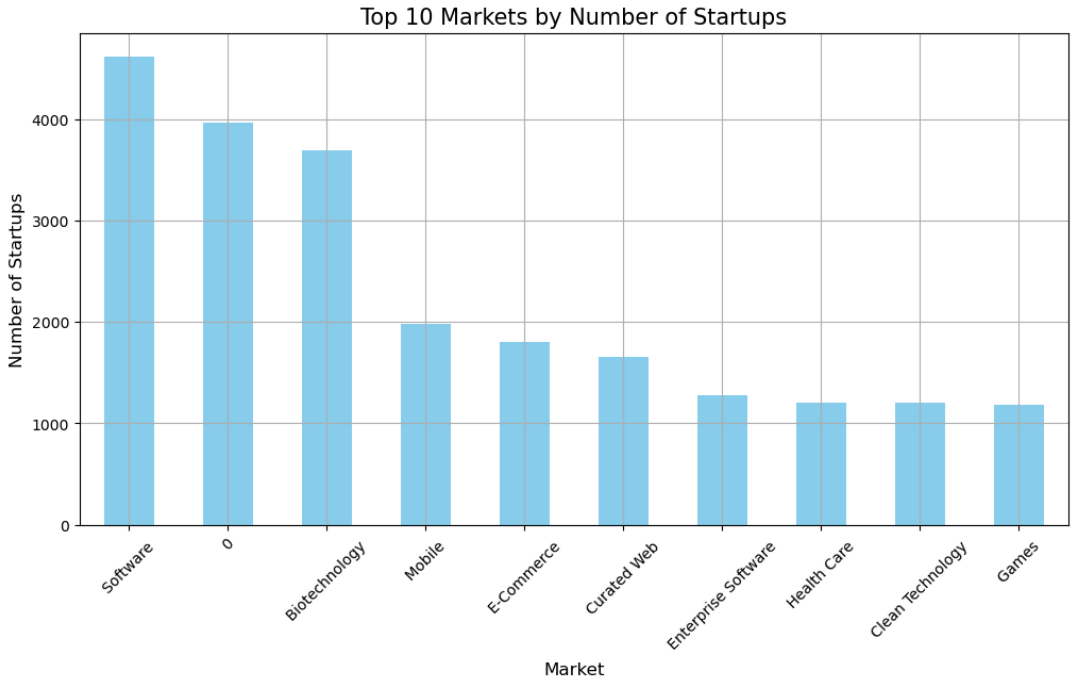
In [153]:
```
# 4. Analyze the distribution of startups by 'market'
market_funding = data1.groupby(' market ')[' funding_total_usd '].sum()
market_startups = data1[' market '].value_counts()
```

In [154]:

```
# 5. Visualize the top 10 markets by total funding
plt.figure(figsize=(12, 6))
market_funding.head(10).plot(kind='bar', color='purple')
plt.title('Top 10 Markets by Total Funding', fontsize=15)
plt.xlabel('Market', fontsize=12)
plt.ylabel('Total Funding (USD)', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



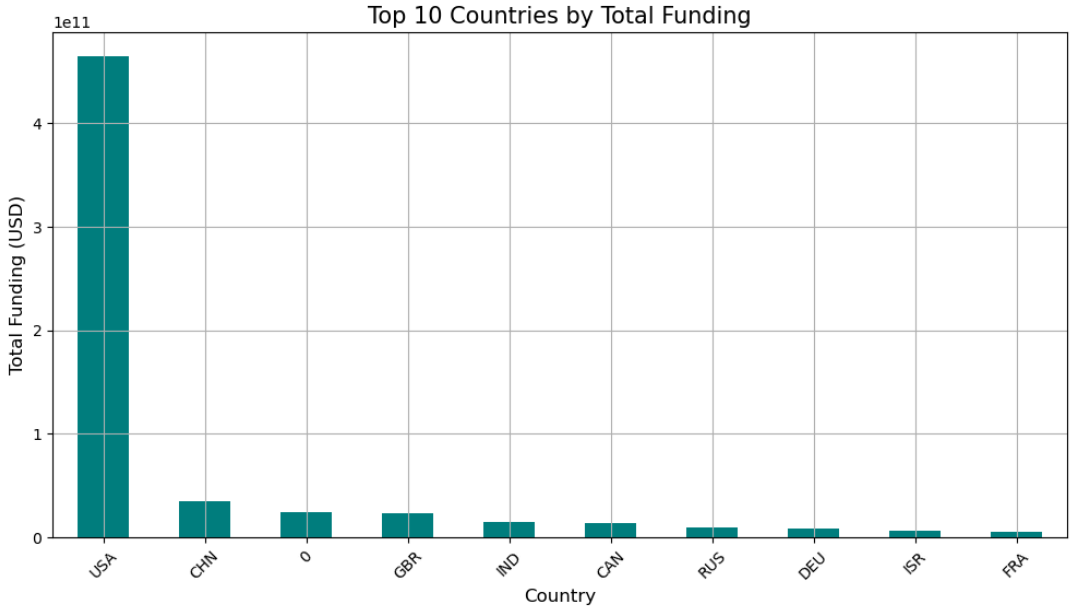
In [155]:
```
# 6. Visualize the top 10 markets by number of startups
plt.figure(figsize=(12, 6))
market_startups.head(10).plot(kind='bar', color='skyblue')
plt.title('Top 10 Markets by Number of Startups', fontsize=15)
plt.xlabel('Market', fontsize=12)
plt.ylabel('Number of Startups', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



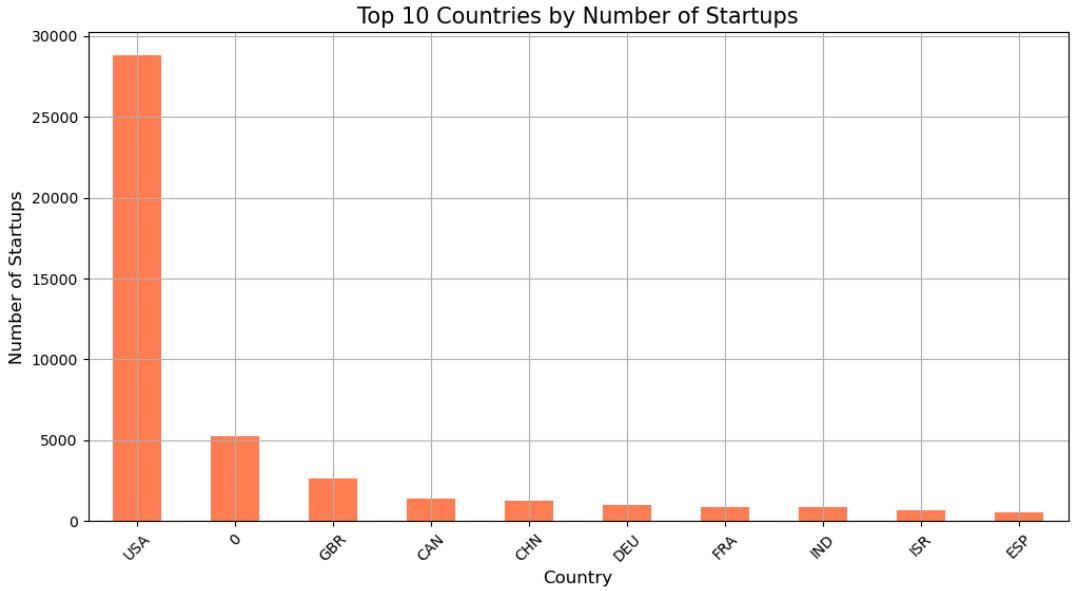
Geographical Insights:

In [156]:
```
# 1. Analyze funding and number of startups by country
country_funding = data1.groupby('country_code')[' funding_total_usd '].
country_startups = data1['country_code'].value_counts()
```

In [157]:
```
# 2. Visualize the top 10 countries by total funding
plt.figure(figsize=(12, 6))
country_funding.head(10).plot(kind='bar', color='teal')
plt.title('Top 10 Countries by Total Funding', fontsize=15)
plt.xlabel('Country', fontsize=12)
plt.ylabel('Total Funding (USD)', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

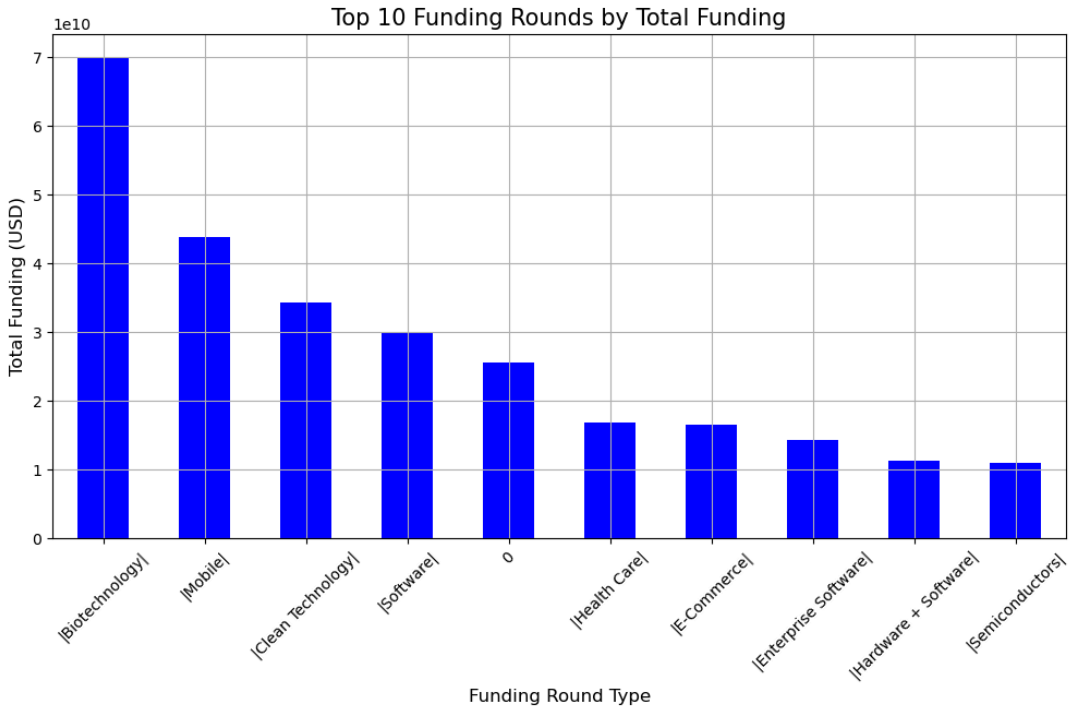


In [158]:
```
# 5. Visualize the top 10 countries by number of startups
plt.figure(figsize=(12, 6))
country_startups.head(10).plot(kind='bar', color='coral')
plt.title('Top 10 Countries by Number of Startups', fontsize=15)
plt.xlabel('Country', fontsize=12)
plt.ylabel('Number of Startups', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

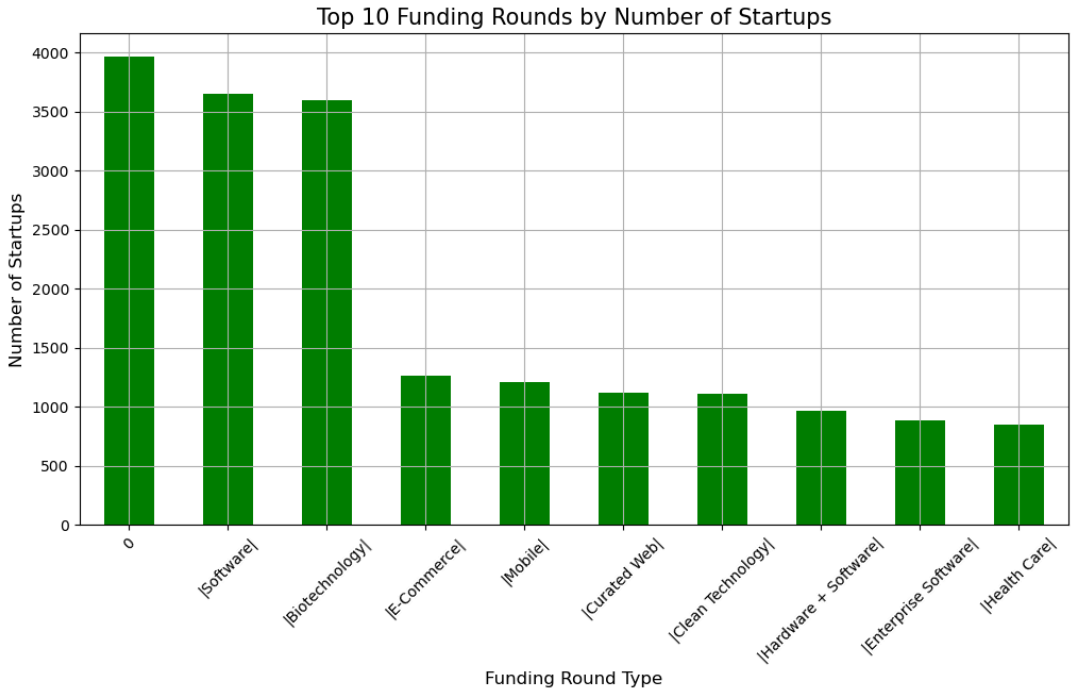


In [159]:
```
# 1. Analyze the total funding by different funding rounds
funding_rounds_funding = data1.groupby('category_list')[' funding_total
funding_rounds_count = data1['category_list'].value_counts()
```

In [160]:
```
# 2. Visualize the top 10 funding rounds by total funding
plt.figure(figsize=(12, 6))
funding_rounds_funding.head(10).plot(kind='bar', color='blue')
plt.title('Top 10 Funding Rounds by Total Funding', fontsize=15)
plt.xlabel('Funding Round Type', fontsize=12)
plt.ylabel('Total Funding (USD)', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



In [161]:
```
# 3. Visualize the top 10 funding rounds by frequency (number of startu
plt.figure(figsize=(12, 6))
funding_rounds_count.head(10).plot(kind='bar', color='green')
plt.title('Top 10 Funding Rounds by Number of Startups', fontsize=15)
plt.xlabel('Funding Round Type', fontsize=12)
plt.ylabel('Number of Startups', fontsize=12)
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



In [162]:
```
from scipy.stats import pearsonr, spearmanr
```

In [163]:
```
## Convert 'funding_total_usd' to numeric
data1[' funding_total_usd '] = pd.to_numeric(data1[' funding_total_usd
```

In [164]:
```
# Drop rows where conversion failed (NaN values)
data = data1.dropna(subset=['funding_total_usd'])
```

In [165]:
```
# Now 'funding_total_usd' is in numeric format
print(data1['funding_total_usd'].head())
```

```
0    1750000.0
1    4000000.0
2      40000.0
3    1500000.0
4      60000.0
Name: funding_total_usd, dtype: float64
```

In [166]:
```
# 1. Compute the correlation matrix
corr_matrix = data1.corr(method='pearson')
```

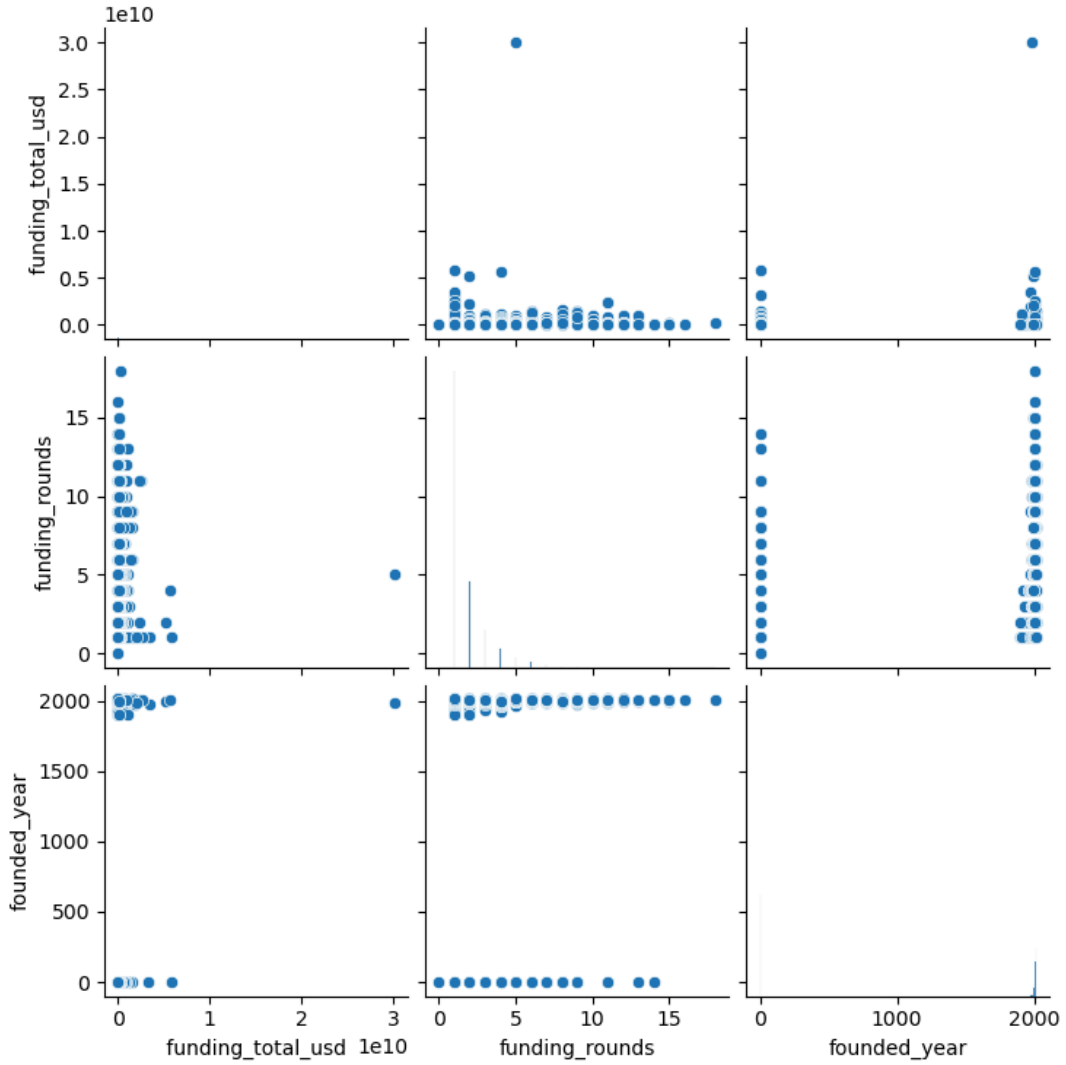
```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[166], line 2
      1 # 1. Compute the correlation matrix
----> 2 corr_matrix = data1.corr(method='pearson')

File ~\Downloads\ANA\Lib\site-packages\pandas\core\frame.py:10054, in DataFrame.corr(self, method, min_periods, numeric_only)
  10052 cols = data.columns
  10053 idx = cols.copy()
> 10054 mat = data.to_numpy(dtype=float, na_value=np.nan, copy=False)
  10056 if method == "pearson":
  10057     correl = libalgos.nancorr(mat, minp=min_periods)

File ~\Downloads\ANA\Lib\site-packages\pandas\core\frame.py:1838, in DataFrame.to_numpy(self, dtype, copy, na_value)
   1836 if dtype is not None:
   1837     dtype = np.dtype(dtype)
-> 1838 result = self._mgr.as_array(dtype=dtype, copy=copy, na_value=na_value)
   1839 if result.dtype is not dtype:
   1840     result = np.array(result, dtype=dtype, copy=False)

File ~\Downloads\ANA\Lib\site-packages\pandas\core\internals\managers.py:1732, in BlockManager.as_array(self, dtype, copy, na_value)
   1730         arr.flags.writeable = False
   1731 else:
-> 1732     arr = self._interleave(dtype=dtype, na_value=na_value)
   1733     # The underlying data was copied within _interleave, so no need
   1734     # to further copy if copy=True or setting na_value
   1736 if na_value is not lib.no_default:

File ~\Downloads\ANA\Lib\site-packages\pandas\core\internals\managers.py:1794, in BlockManager._interleave(self, dtype, na_value)
   1792     else:
   1793         arr = blk.get_values(dtype)
-> 1794     result[rl.indexer] = arr
   1795     itemmask[rl.indexer] = 1
   1797 if not itemmask.all():

ValueError: could not convert string to float: '#waywire'
```

In [167]:
```
# 3. Analyze specific correlations
## Correlation between 'funding_total_usd' and 'funding_rounds'
funding_total = data1[' funding_total_usd ']
funding_rounds = data1['funding_rounds']
corr_coefficient, p_value = pearsonr(funding_total, funding_rounds)
print(f"Pearson Correlation between 'funding_total_usd' and 'funding_ro
```

```
Pearson Correlation between 'funding_total_usd' and 'funding_rounds':
0.11 (p-value: 5.33e-134)
```

In [168]:
```
# 4. Visualize relationships with pair plots
sns.pairplot(data1[[' funding_total_usd ', 'funding_rounds', 'founded_y
plt.show()
```

```
C:\Users\Dell\Downloads\ANA\Lib\site-packages\seaborn\axisgrid.py:118:
UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```

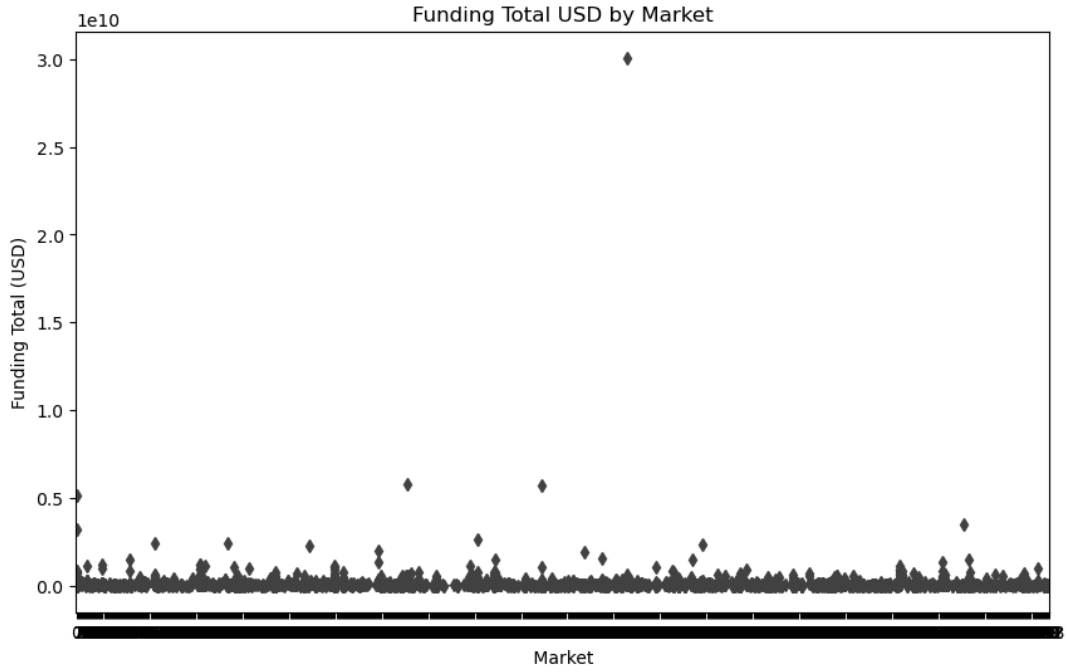


In [169]:
```
# 5. Correlation between categorical and continuous variables
## Encode 'market' as a categorical variable
data1[' market '] = data1[' market '].astype('category').cat.codes
```

In [170]:
```
## Compute correlation
market_corr = data1[[' market ', ' funding_total_usd ']].corr(method='p
print(f"Correlation between ' market ' and 'funding_total_usd':\n{marke
```

```
Correlation between ' market ' and 'funding_total_usd':
                     market    funding_total_usd
 market              1.000000          -0.001054
 funding_total_usd  -0.001054           1.000000
```

In [171]:
```
# 6. Visualize the relationship between 'market' and 'funding_total_usd
plt.figure(figsize=(10, 6))
sns.boxplot(x=' market ', y=' funding_total_usd ', data=data1)
plt.title('Funding Total USD by Market')
plt.xlabel(' Market ')
plt.ylabel('Funding Total (USD)')
plt.show()
```

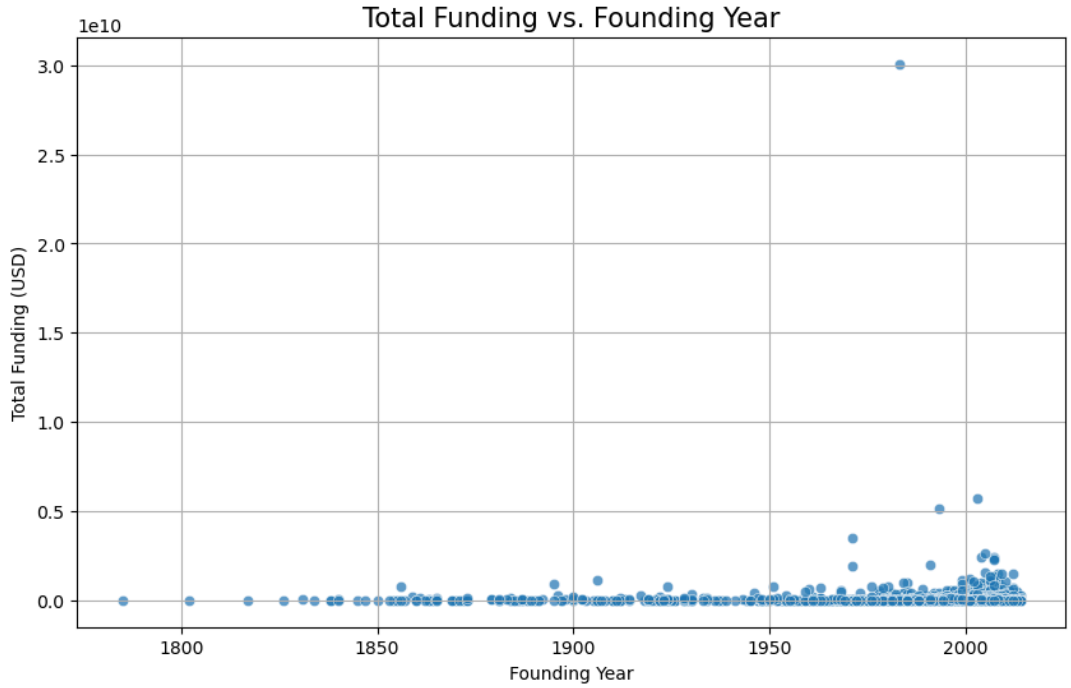


In [172]:
```
# Convert 'founded_at' and 'first_funding_at' to datetime
data1['founded_at'] = pd.to_datetime(data1['founded_at'], errors='coerc
data1['first_funding_at'] = pd.to_datetime(data1['first_funding_at'], e
```

In [173]:
```
# Extract the founding year
data1['founded_year'] = data1['founded_at'].dt.year
```

In [174]:
```
# 1. Correlation between founding year and total funding
founding_vs_funding = data1[['founded_year', 'funding_total_usd']].drop
```

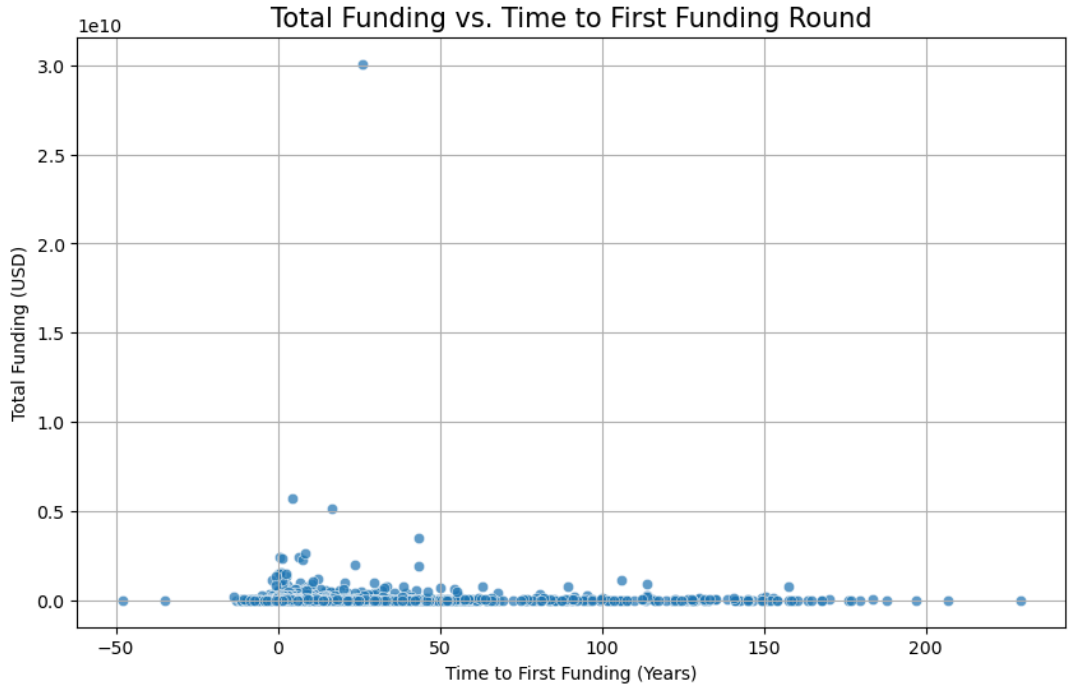
In [179]:
```
# Plotting the relationship between founding year and total funding
plt.figure(figsize=(10, 6))
sns.scatterplot(x='founded_year', y='funding_total_usd', data=founding_
plt.title('Total Funding vs. Founding Year', fontsize=15)
plt.xlabel('Founding Year')
plt.ylabel('Total Funding (USD)')
plt.grid(True)
plt.show()
```



In [180]:
```
# 2. Time to first funding round (years)
# Calculate the number of days between founding and first funding
data1['time_to_first_funding'] = (data1['first_funding_at'] - data1['fo
```

In [181]:
```
# 3. Correlation between time to first funding and total funding
timing_vs_funding = data1[['time_to_first_funding', 'funding_total_usd'
```

In [182]:
```
# Plotting the relationship between time to first funding and total fun
plt.figure(figsize=(10, 6))
sns.scatterplot(x='time_to_first_funding', y='funding_total_usd', data=
plt.title('Total Funding vs. Time to First Funding Round', fontsize=15)
plt.xlabel('Time to First Funding (Years)')
plt.ylabel('Total Funding (USD)')
plt.grid(True)
plt.show()
```



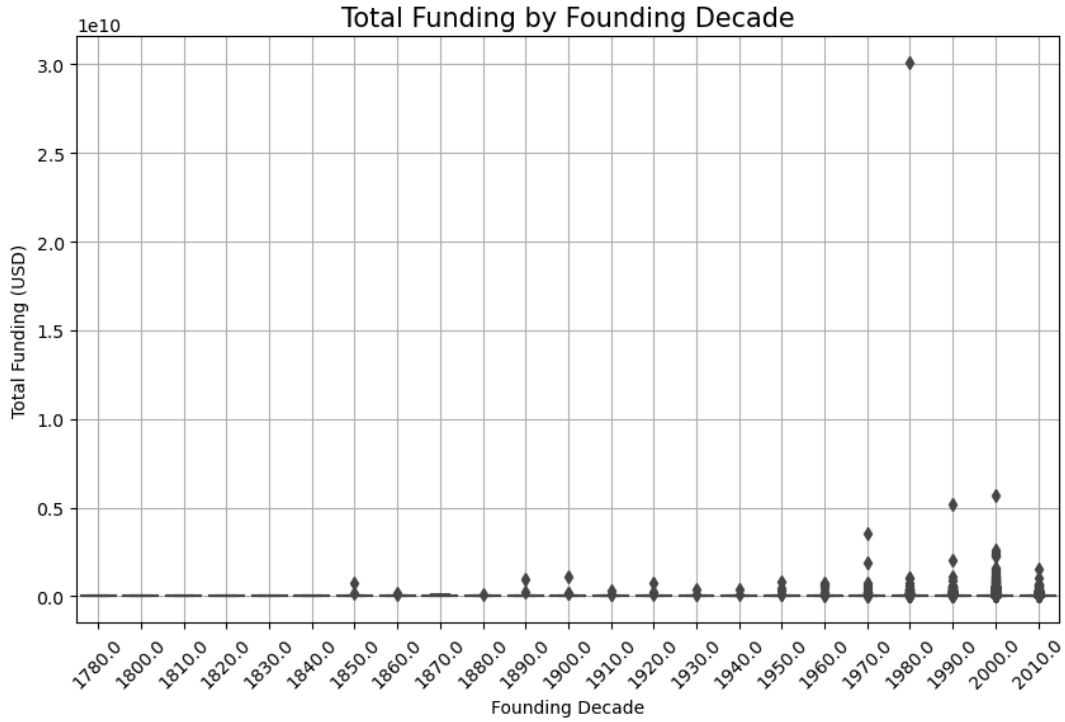
In [183]:
```
# 4. Analyze the correlation values
corr_founded_year_funding = founding_vs_funding['founded_year'].corr(fo
corr_time_to_funding_funding = timing_vs_funding['time_to_first_funding
```

In [184]:
```
print(f"Correlation between Founding Year and Total Funding: {corr_foun
print(f"Correlation between Time to First Funding and Total Funding: {c
```

```
Correlation between Founding Year and Total Funding: -0.06
Correlation between Time to First Funding and Total Funding: 0.05
```

In [185]:
```
#Boxplot to show distribution of total funding by founding decade
data1['founded_decade'] = (data1['founded_year'] // 10) * 10
```

In [186]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='founded_decade', y='funding_total_usd', data=data1)
plt.title('Total Funding by Founding Decade', fontsize=15)
plt.xlabel('Founding Decade')
plt.ylabel('Total Funding (USD)')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



In [187]:
```
import scipy.stats as stats
```

In [188]:
```
# 1. Hypothesis 1: Do startups in certain categories receive significan
# ------------------------------------------------------------------------
## ANOVA test: Funding by Category
# Group funding by category
categories = data1['category_list'].unique()
funding_by_category = [data1[data1['category_list'] == cat]['funding_to
```

In [189]:
```
# Perform one-way ANOVA
anova_result = stats.f_oneway(*funding_by_category)
```

In [190]:
```
# Check p-value
if anova_result.pvalue < 0.05:
    print("Reject the null hypothesis: There's a significant difference
else:
    print("Fail to reject the null hypothesis: No significant differenc
```

```
Fail to reject the null hypothesis: No significant difference in fundi
ng across categories.
```

In [191]:
```
# 2. Hypothesis 2: Does location (e.g., country) significantly impact t
# ------------------------------------------------------------------------
## T-test: Funding by Location (for two countries, say 'USA' vs. 'Canad
```

In [192]:
```
# Drop rows with missing country information
data1.dropna(subset=['country_code'], inplace=True)
```

In [193]:
```
# Filter for two specific countries (e.g., USA and Canada)
usa_funding = data1[data1['country_code'] == 'USA']['funding_total_usd'
canada_funding = data1[data1['country_code'] == 'CAN']['funding_total_u
```

In [194]:
```
# Perform independent t-test
t_stat, p_val = stats.ttest_ind(usa_funding, canada_funding, nan_policy
```

In [195]:
```
# Check p-value
if p_val < 0.05:
    print("Reject the null hypothesis: There's a significant difference
else:
    print("Fail to reject the null hypothesis: No significant differenc
```

```
Fail to reject the null hypothesis: No significant difference in fundi
ng between USA and Canada.
```

In [196]:
```
## T-test: Funding by Location (for two countries, say 'USA' vs. 'China
```

In [197]:
```
# Filter for two specific countries (e.g., USA and China)
usa_funding = data1[data1['country_code'] == 'USA']['funding_total_usd'
china_funding = data1[data1['country_code'] == 'CHN']['funding_total_us
```

In [198]:
```
# Perform independent t-test
t_stat, p_val = stats.ttest_ind(usa_funding, china_funding, nan_policy=
```

In [199]:
```
# Check p-value
if p_val < 0.05:
    print("Reject the null hypothesis: There's a significant difference
else:
    print("Fail to reject the null hypothesis: No significant differenc
```

```
Reject the null hypothesis: There's a significant difference in fundin
g between USA and China.
```

In [200]:
```
## T-test: Funding by Location (for two countries, say 'India' vs. 'Chi
```

In [201]:
```
# Filter for two specific countries (e.g., India and China)
india_funding = data1[data1['country_code'] == 'IND']['funding_total_us
china_funding = data1[data1['country_code'] == 'CHN']['funding_total_us
```

In [202]:
```
# Perform independent t-test
t_stat, p_val = stats.ttest_ind(india_funding, china_funding, nan_polic
```

In [203]:
```
# Check p-value
if p_val < 0.05:
    print("Reject the null hypothesis: There's a significant difference
else:
    print("Fail to reject the null hypothesis: No significant differenc
```

```
Reject the null hypothesis: There's a significant difference in fundin
g between India and China.
```

In [204]:
```
# Testing Hypothesis
```

In [205]:
```
# Drop rows with missing funding_total_usd or category
data1.dropna(subset=['funding_total_usd', 'category_list', 'country_cod
```

In [206]:
```
### 1. Hypothesis Test 1: Mean Funding Across Market Categories (ANOVA)
# ----------------------------------------------------------------------
# Group funding by market category
categories = data1['category_list'].unique()
funding_by_category = [data1[data1['category_list'] == cat]['funding_to
```

In [207]:
```
# Perform one-way ANOVA test
anova_result = stats.f_oneway(*funding_by_category)
```

In [208]:
```
# Check p-value
if anova_result.pvalue < 0.05:
    print("Reject the null hypothesis: There's a significant difference
else:
    print("Fail to reject the null hypothesis: No significant differenc
```

```
Fail to reject the null hypothesis: No significant difference in mean
funding across market categories.
```

In [209]:
```
### 2. Hypothesis Test 2: Mean Funding Across Geographical Regions (t-T
# ----------------------------------------------------------------------
# Compare mean funding between two countries, e.g., USA vs. China
```

In [210]:
```
# Filter for USA and China
usa_funding = data1[data1['country_code'] == 'USA']['funding_total_usd'
china_funding = data1[data1['country_code'] == 'CHN']['funding_total_us
```

In [211]:
```
# Perform independent t-test
t_stat, p_val = stats.ttest_ind(usa_funding, china_funding, nan_policy=
```

In [212]:
```
# Check p-value
if p_val < 0.05:
    print("Reject the null hypothesis: There's a significant difference
else:
    print("Fail to reject the null hypothesis: No significant differenc
```

```
Reject the null hypothesis: There's a significant difference in mean f
unding between USA and China.
```

In [213]:
```
### 3. Hypothesis Test 2: Mean Funding Across Geographical Regions (t-T
# ----------------------------------------------------------------------
# Compare mean funding between two countries, e.g., India vs. China
```

In [214]:
```
# Filter for USA and China
india_funding = data1[data1['country_code'] == 'IND']['funding_total_us
china_funding = data1[data1['country_code'] == 'CHN']['funding_total_us
```

In [215]:
```
# Perform independent t-test
t_stat, p_val = stats.ttest_ind(india_funding, china_funding, nan_polic
```

In [216]:
```
# Check p-value
if p_val < 0.05:
    print("Reject the null hypothesis: There's a significant difference
else:
    print("Fail to reject the null hypothesis: No significant differenc
```

```
Reject the null hypothesis: There's a significant difference in mean f
unding between India and China.
```

In [217]:
```
### Multiple Countries (ANOVA for More than Two Countries)
# ----------------------------------------------------------------------
# Compare mean funding across multiple countries using ANOVA
countries = ['USA', 'Canada', 'UK']  # Add more countries as needed
funding_by_country = [data1[data1['country_code'] == country_code]['fun
```

In [218]:
```
# Perform ANOVA
anova_country_result = stats.f_oneway(*funding_by_country)
```

```
C:\Users\Dell\Downloads\ANA\Lib\site-packages\scipy\stats\_stats_py.p
y:4133: DegenerateDataWarning: at least one input has length 0
  warnings.warn(stats.DegenerateDataWarning('at least one input '
```

In [219]:
```
# Check p-value for country comparison
if anova_country_result.pvalue < 0.05:
    print("Reject the null hypothesis: There's a significant difference
else:
    print("Fail to reject the null hypothesis: No significant differenc
```

```
Fail to reject the null hypothesis: No significant difference in mean
funding across countries.
```

In [ ]:

In [ ]: