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

In [3]:
```
# Load Data

df = pd.read_csv("shark_tank_dataset.csv")
df
```

Out[3]:

	startup_name	domain	funding_amount	equity_offered	founders	stage
0	InnoHub_0	Fashion	NaN	unknown	Unknown	Series B
1	CloudLink_1	logistics	9839725.0	NaN	3 founders	Angel
2	GreenHub_2	Agri	NaN	NaN	2 founders	Series C
3	UrbanBridge_3	AI	NaN	NaN	2 founders	Seed
4	BrightMart_4	Tech	NaN	unknown	3 founders	Series C
...	...	...	...	...	...	...
4995	AISystems_4995	Health	NaN	unknown	NaN	Series B
4996	MetaGen_4996	food	96965966.0	NaN	NaN	Bootstrapped
4997	GreenBox_4997	Logistics	NaN	11.55%	NaN	Series B
4998	MetaKart_4998	edtech	11280488.0	unknown	NaN	Pre-Seed
4999	FreshWave_4999	AI	NaN	NaN	2 founders	Pre-Seed

5000 rows × 10 columns

Data Exploration

- It helps data scientists understand the dataset, identify patterns, and gain insights before further analysis

In [5]:
```
# It represent the number of rows and columns in the DataFrame
df.shape
```

Out[5]:
```
(5000, 10)
```

In [6]:

```
# To extract the column names of a DataFrame
df.columns.tolist()
```

Out[6]:

```
['startup_name',
 'domain',
 'funding_amount',
 'equity_offered',
 'founders',
 'stage',
 'country',
 'season',
 'deal_status',
 'investor']
```

In [7]:

```
# Prints information about the DataFrame
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 10 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   startup_name    5000 non-null   object 
 1   domain          4767 non-null   object 
 2   funding_amount  1299 non-null   float64
 3   equity_offered  2500 non-null   object 
 4   founders        3524 non-null   object 
 5   stage           4369 non-null   object 
 6   country         4500 non-null   object 
 7   season          3867 non-null   float64
 8   deal_status     4178 non-null   object 
 9   investor        4546 non-null   object 
dtypes: float64(2), object(8)
memory usage: 390.8+ KB
```

Unique---> **Returns unique values from a data series**

In [14]:

```
#Categorical

df["startup_name"].unique
```

Out[14]:

```
<bound method Series.unique of 0            InnoHub_0
1           CloudLink_1
2            GreenHub_2
3         UrbanBridge_3
4          BrightMart_4
             ...      
4995    AISystems_4995
4996      MetaGen_4996
4997     GreenBox_4997
4998     MetaKart_4998
4999    FreshWave_4999
Name: startup_name, Length: 5000, dtype: object>
```

In [16]:

```
df["domain"].unique
```

```
Out[16]: <bound method Series.unique of 0         Fashion
1       logistics
2            Agri
3              AI
4            Tech
          ...    
4995       Health
4996         food
4997    Logistics
4998       edtech
4999           AI
Name: domain, Length: 5000, dtype: object>
```

```
In [18]: df["domain"] = df["domain"].fillna("Unknown")
```

```
In [20]: #Categorical

         df["domain"].unique
```

```
Out[20]: <bound method Series.unique of 0         Fashion
1       logistics
2            Agri
3              AI
4            Tech
          ...    
4995       Health
4996         food
4997    Logistics
4998       edtech
4999           AI
Name: domain, Length: 5000, dtype: object>
```

```
In [22]: #continous

         df["funding_amount"].unique()
```

```
Out[22]: array([      nan,  9839725., 31867562., ..., 15058242., 96965966.,
       11280488.])
```

.replace is replaces whole values works with strings, numbers, lists, etc (all 25 value is changed in 39)

str.replace is works on entire column

- (mr rahul ---> rahul,
- mr dev ----> dev,
- mr sonu ----> sonu)

```
In [25]: df["equity_offered"].unique
```

Out[25]:

```
<bound method Series.unique of 0       unknown
1           NaN
2           NaN
3           NaN
4       unknown
         ...
4995    unknown
4996        NaN
4997     11.55%
4998    unknown
4999        NaN
Name: equity_offered, Length: 5000, dtype: object>
```

Unknown” is for text (categorical) data.

NaN is for number (continuous) data.

In [28]:

```
# replace "unknown", "", "N/A" with nan

df["equity_offered"] = df["equity_offered"].replace(["unknown", "", "N/A"], np.nan)

# "unknown", empty "", and "N/A" and replace with nan (missing value).
```

In [30]:

```
# Replace % to " "
# regex=False means treat % as a normal character
# errors="coerce" means if conversion fails, set it to NaN.

df["equity_offered"] = df["equity_offered"].str.replace("%", "", regex=False)  # re
df["equity_offered"] = pd.to_numeric(df["equity_offered"], errors="coerce") # Conve

# pd.to_numeric(..., errors="coerce")  ..........>  Convert to numbers(floats), in
```

In [32]:

```
# Continous

df["equity_offered"].unique()[:20]
```

Out[32]:

```
array([  nan,  4.61, 49.58, 48.04, 48.71, 15.49, 25.97, 38.2 , 29.68,
       38.79, 47.35,  3.28, 39.86, 16.29, 11.39, 32.41,  8.64,  2.85,
       20.41, 32.66])
```

In [33]:

```
df.loc[df["deal_status"] == "Rejected", "equity_offered"] = 0
```

In [36]:

```
df["founders"].unique()
```

Out[36]:

```
array(['Unknown', '3 founders', '2 founders', '4 founders', nan,
       '1 founder'], dtype=object)
```

In [38]:

```
# Replace variations of unknown with NaN
df["founders"] = df["founders"].replace(["Unknown", "unknown", ""], np.nan)

# Extract the number from strings like "3 founders", "1 founder"
df["founders"] = df["founders"].str.extract(r'(\d+)').astype(float)
```

In [40]:

```
df["founders"].unique()
```

```
Out[40]: array([nan,  3.,  2.,  4.,  1.])
```

```
In [42]: df["founders"] = df["founders"].astype("Int64")  # Pandas integer that supports NaN
```

```
In [44]: # count

         df["founders"].unique()
```

```
Out[44]: <IntegerArray>
         [<NA>, 3, 2, 4, 1]
         Length: 5, dtype: Int64
```

```
In [46]: df["stage"].unique()
```

```
Out[46]: array(['Series B', 'Angel', 'Series C', 'Seed', 'Bootstrapped',
                'Pre-Seed', 'Series A', nan], dtype=object)
```

```
In [48]: stage_order = [
             "Bootstrapped",
             "Pre-seed",
             "Seed",
             "Angel",
             "Series A",
             "Series B",
             "Series C"
         ]

         df["stage"] = pd.Categorical(df["stage"], categories=stage_order, ordered=True)
```

Always replace NaN with "Unknown" for categorical columns before doing any crosstab or grouping — otherwise pandas will count nothing.

```
In [51]: df["stage"].unique()
```

```
Out[51]: ['Series B', 'Angel', 'Series C', 'Seed', 'Bootstrapped', NaN, 'Series A']
         Categories (7, object): ['Bootstrapped' < 'Pre-seed' < 'Seed' < 'Angel' < 'Series
         A' < 'Series B' < 'Series C']
```

```
In [53]: df["stage"] = df["stage"].cat.add_categories("Unknown")
         df["stage"] = df["stage"].fillna("Unknown")
```

```
In [55]: # Categorical

         df["stage"].unique()
```

```
Out[55]: ['Series B', 'Angel', 'Series C', 'Seed', 'Bootstrapped', 'Unknown', 'Series A']
         Categories (8, object): ['Bootstrapped' < 'Pre-seed' < 'Seed' < 'Angel' < 'Series
         A' < 'Series B' < 'Series C' < 'Unknown']
```

In [57]:

```
df["country"].unique()
```

Out[57]:

```
array([nan, 'USA', 'uae', 'india', 'India', 'canada', 'usa', 'Canada',
       'UAE', 'uk'], dtype=object)
```

In [59]:

```
df["country"] = df["country"].str.lower()

df["country"] = df["country"].replace({
    "india": "India",
    "usa": "USA",
    "canada": "Canada",
    "uae": "UAE",
    "uk": "UK",
    "": np.nan
})

df["country"] = df["country"].fillna("Unknown")
```

In [61]:

```
# Categorical

df["country"].unique()
```

Out[61]:

```
array(['Unknown', 'USA', 'UAE', 'India', 'Canada', 'UK'], dtype=object)
```

In [63]:

```
df["season"].unique()
```

Out[63]:

```
array([nan,  6.,  7.,  3.,  5.,  1.,  2.,  4.])
```

In [65]:

```
df["season"] = df["season"].astype("Int64")  # Pandas integer that supports NaN
```

In [67]:

```
# count

df["season"].unique()
```

Out[67]:

```
<IntegerArray>
[<NA>, 6, 7, 3, 5, 1, 2, 4]
Length: 8, dtype: Int64
```

In [69]:

```
df["deal_status"].unique()
```

Out[69]:

```
array(['Funded', 'No Deal', 'Partially Funded', 'Offer Withdrawn', nan,
       'Rejected'], dtype=object)
```

In [71]:

```
df["deal_status"] = df["deal_status"].fillna("Unknown")
```

In [73]:

```
# Categorical

df["deal_status"].unique()
```

Out[73]:

```
array(['Funded', 'No Deal', 'Partially Funded', 'Offer Withdrawn',
       'Unknown', 'Rejected'], dtype=object)
```

In [75]:

```
# Categorical
```

```
df["investor"].unique()
```

Out[75]:

```
array(['Namita', 'Lori', 'Barbara', 'Peyush', nan, 'Aman', 'Ashneer',
       'Anupam', 'Vineeta', 'Kevin', 'Mark'], dtype=object)
```

In [77]:

```
df["investor"] = df["investor"].fillna("Unknown")
```

In [79]:

```
# Categorical

df["investor"].unique()
```

Out[79]:

```
array(['Namita', 'Lori', 'Barbara', 'Peyush', 'Unknown', 'Aman',
       'Ashneer', 'Anupam', 'Vineeta', 'Kevin', 'Mark'], dtype=object)
```

To check which is continous count And discrete

In [82]:

```
continous = ["funding_amount","equity_offered"]

# Count means No.of
count = ["founders","season"]

categorical = ["startup_name","domain","stage","country","deal_status","investor"]
```

Generate descriptive statistics of a DataFrame

In [85]:

```
df[continous].describe()
```

Out[85]:

	funding_amount	equity_offered
count	1.299000e+03	1896.000000
mean	4.989087e+07	17.073713
std	2.928822e+07	19.696373
min	1.908900e+04	0.000000
25%	2.349381e+07	0.000000
50%	4.911468e+07	7.295000
75%	7.619417e+07	33.407500
max	9.996534e+07	59.890000

In [87]:

```
pd.set_option('display.float_format', '{:,.0f}'.format)
```

In [89]:

```
df[continous].describe()
```

Out[89]:

	funding_amount	equity_offered
count	1,299	1,896
mean	49,890,874	17
std	29,288,218	20
min	19,089	0
25%	23,493,808	0
50%	49,114,685	7
75%	76,194,172	33
max	99,965,336	60

In [91]:

```
df[count].describe()
```

Out[91]:

	founders	season
count	2,798	3,867
mean	2	4
std	1	2
min	1	1
25%	1	2
50%	2	4
75%	3	6
max	4	7

Top means Most common category

Freq means this is the number of times the top value appears

In [94]:

```
df[categorical].describe()
```

Out[94]:

	startup_name	domain	stage	country	deal_status	investor
count	5000	5000	5000	5000	5000	5000
unique	5000	23	7	6	6	11
top	InnoHub_0	Agri	Unknown	India	Offer Withdrawn	Barbara
freq	1	248	1289	1033	881	490

In [96]:

```
# Check Missing value
```

```
df.isnull().sum()
```

Out[96]:

```
startup_name         0
domain               0
funding_amount    3701
equity_offered    3104
founders          2202
stage                0
country              0
season            1133
deal_status          0
investor             0
dtype: int64
```

Treat Missing values (NAN)

In [99]:

```
# Fill numeric columns logically
df["funding_amount"].fillna(df["funding_amount"].median(), inplace=True)
df["equity_offered"].fillna(df["equity_offered"].median(), inplace=True)
df["founders"].fillna(df["founders"].median(), inplace=True)
df["season"].fillna(df["season"].mode()[0], inplace=True)
```

```
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_19244\1597932572.py:2: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained assig
nment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df["funding_amount"].fillna(df["funding_amount"].median(), inplace=True)
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_19244\1597932572.py:3: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained assig
nment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df["equity_offered"].fillna(df["equity_offered"].median(), inplace=True)
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_19244\1597932572.py:4: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained assig
nment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df["founders"].fillna(df["founders"].median(), inplace=True)
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_19244\1597932572.py:5: FutureWarning:
A value is trying to be set on a copy of a DataFrame or Series through chained assig
nment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because
the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method
({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform
the operation inplace on the original object.


  df["season"].fillna(df["season"].mode()[0], inplace=True)
```

In [101…

```
# Final check
df.isnull().sum()
```

Out[101…

```
startup_name      0
domain            0
funding_amount    0
equity_offered    0
founders          0
stage             0
country           0
season            0
deal_status       0
investor          0
dtype: int64
```

skewness is only meaningful for numerical (continous or count) variables

In [104…

```
# calculates the skew for each column

df[continous].skew()
```

Out[104…

```
funding_amount   0
equity_offered   2
dtype: float64
```

In [106…

```
df[count].skew()
```

Out[106…

```
founders   1
season     1
dtype: Float64
```

In [108…

```
# To check duplicate

df.duplicated().sum()
```

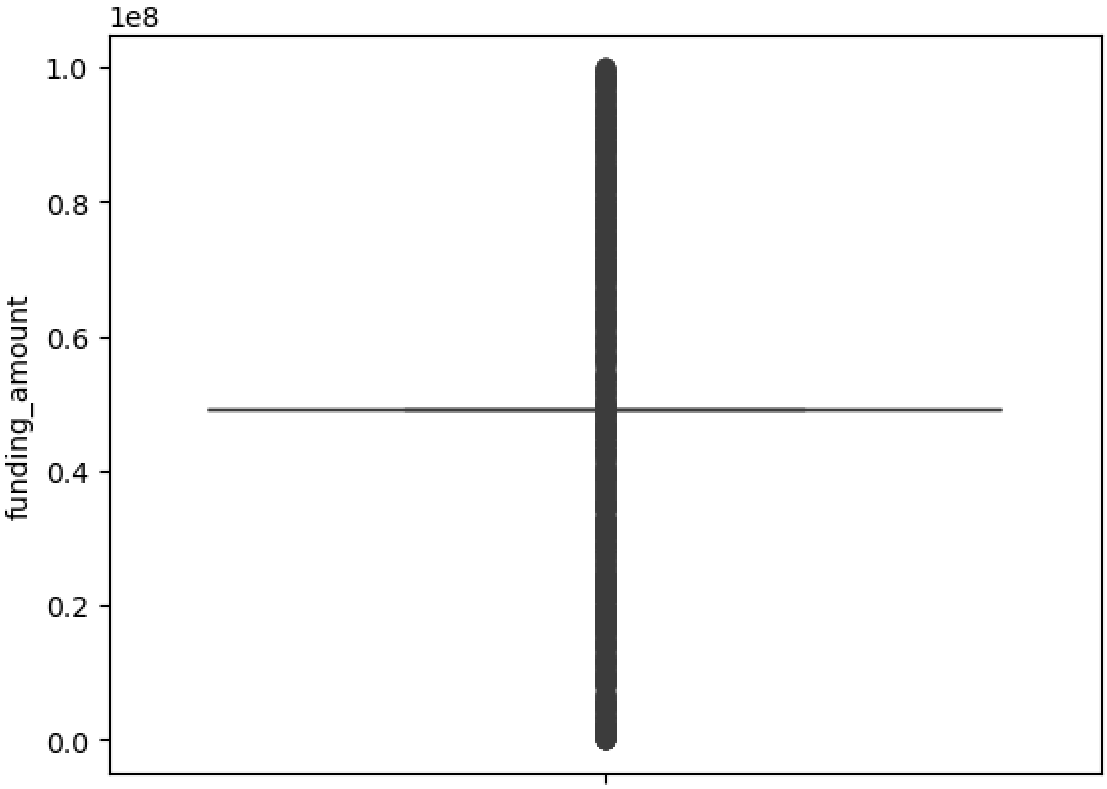
Out[108…

```
0
```

To check Outlier

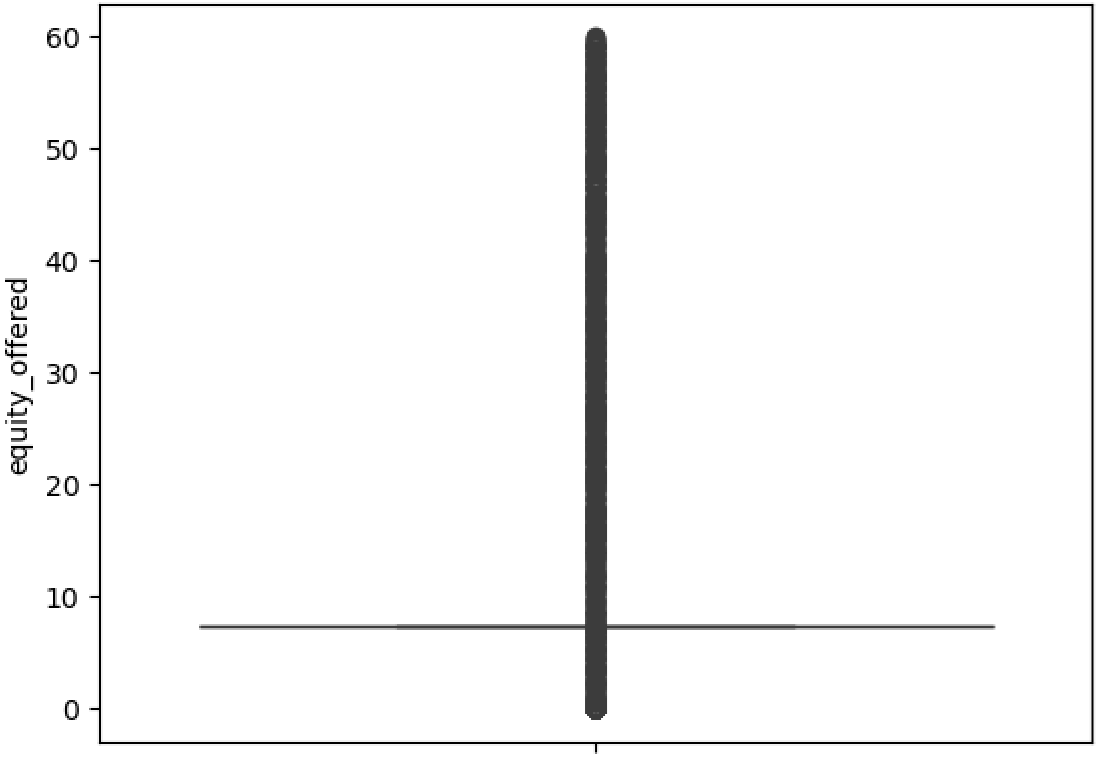
In [111…

```
sns.boxplot(df["funding_amount"])
plt.show()
```



In [113…

```
sns.boxplot(df["equity_offered"])
plt.show()
```



Data Cleaning

Retrain the outliers (Keep them as it is) this is genuine data

In [116…

```
# clean data

df
```

Out[116…

	startup_name	domain	funding_amount	equity_offered	founders	stage
0	InnoHub_0	Fashion	49,114,685	7	2	Series B
1	CloudLink_1	logistics	9,839,725	7	3	Angel
2	GreenHub_2	Agri	49,114,685	7	2	Series C
3	UrbanBridge_3	AI	49,114,685	7	2	Seed
4	BrightMart_4	Tech	49,114,685	7	3	Series C
...	...	...	...	...	...	...
4995	AISystems_4995	Health	49,114,685	7	2	Series B
4996	MetaGen_4996	food	96,965,966	0	2	Bootstrapped
4997	GreenBox_4997	Logistics	49,114,685	12	2	Series B
4998	MetaKart_4998	edtech	11,280,488	7	2	Unknown
4999	FreshWave_4999	AI	49,114,685	0	2	Unknown

5000 rows × 10 columns

In [118…

```
df.head(10)
```

Out[118…

	startup_name	domain	funding_amount	equity_offered	founders	stage	co
0	InnoHub_0	Fashion	49,114,685	7	2	Series B	Unk
1	CloudLink_1	logistics	9,839,725	7	3	Angel	
2	GreenHub_2	Agri	49,114,685	7	2	Series C	
3	UrbanBridge_3	AI	49,114,685	7	2	Seed	
4	BrightMart_4	Tech	49,114,685	7	3	Series C	
5	MetaSolutions_5	Tech	49,114,685	5	4	Bootstrapped	
6	SmartNest_6	fashion	49,114,685	7	2	Unknown	
7	FreshNest_7	fintech	49,114,685	7	2	Series A	
8	NextGrow_8	ai	49,114,685	50	3	Seed	
9	NextBridge_9	Unknown	49,114,685	48	2	Unknown	

In [120…

```
df.tail(10)
```

Out[120…

	startup_name	domain	funding_amount	equity_offered	founders	stag
4990	QuantumEdge_4990	tech	49,114,685	0	4	Series
4991	FinGen_4991	edtech	85,870,599	7	2	Unknow
4992	FreshFoods_4992	food	15,058,242	7	1	See
4993	MetaPulse_4993	EdTech	49,114,685	26	2	Series
4994	SmartKart_4994	ai	49,114,685	0	2	Series
4995	AISystems_4995	Health	49,114,685	7	2	Series
4996	MetaGen_4996	food	96,965,966	0	2	Bootstrappe
4997	GreenBox_4997	Logistics	49,114,685	12	2	Series
4998	MetaKart_4998	edtech	11,280,488	7	2	Unknow
4999	FreshWave_4999	AI	49,114,685	0	2	Unknow

Export DataFrames to CSV

- After cleaned data you can used in PowerBI or Tableau

```
In [123…  df.to_csv("sharktank_clean_dataset.csv",index=False)
```

Data Analysis

- Measures + Plots
- Univariate, Bivariate, Multivariate

Applying various questions or logics on dataset

- value_counts() = To count the occurrences of each unique value within a specific column (or Series) of a Pandas DataFrame.
- describe() = Generates descriptive statistics of a DataFrame

Univariate Measures + Plots

Measures

For continous

```
In [129…  df["funding_amount"].describe()
```

```
Out[129…  count        5,000
          mean    49,316,339
          std     14,928,001
          min         19,089
          25%     49,114,685
          50%     49,114,685
          75%     49,114,685
          max     99,965,336
          Name: funding_amount, dtype: float64
```

```
In [131…  df["equity_offered"].describe()
```

```
Out[131…  count   5,000
          mean       11
          std        13
          min         0
          25%         7
          50%         7
          75%         7
          max        60
          Name: equity_offered, dtype: float64
```

For count

```
In [134…  df["founders"].value_counts()
```

Out[134…

```
founders
2    2908
1     702
3     700
4     690
Name: count, dtype: Int64
```

In [136…

```
df["season"].value_counts()
```

Out[136…

```
season
2    1733
3     573
6     558
1     542
4     538
7     534
5     522
Name: count, dtype: Int64
```

FOR CATEGORICAL

In [139…

```
df["startup_name"].value_counts()
```

Out[139…

```
startup_name
InnoHub_0             1
FreshNest_3330        1
HealthGen_3337        1
FoodSolutions_3336    1
GreenFoods_3335       1
                     ..
NextHub_1666          1
AgriBridge_1665       1
MetaKart_1664         1
QuantumFoods_1663     1
FreshWave_4999        1
Name: count, Length: 5000, dtype: int64
```

In [141…

```
df["domain"].value_counts()
```

```
Out[141…   domain
           Agri          248
           Tech          245
           tech          242
           Unknown       233
           Fashion       226
           Travel        225
           fintech       224
           Food          224
           ecommerce     218
           food          217
           ai            214
           agri          214
           AI            214
           E-commerce    214
           logistics     212
           Logistics     212
           health        209
           travel        207
           FinTech       205
           edtech        203
           Health        202
           EdTech        200
           fashion       192
           Name: count, dtype: int64
```

```
In [143…   df["stage"].value_counts()
```

```
Out[143…   stage
           Unknown        1289
           Series B        662
           Angel           638
           Bootstrapped    623
           Series A        614
           Seed            595
           Series C        579
           Pre-seed          0
           Name: count, dtype: int64
```

```
In [145…   df["country"].value_counts()
```

```
Out[145…   country
           India      1033
           Canada     1004
           UAE         997
           USA         989
           Unknown     500
           UK          477
           Name: count, dtype: int64
```

```
In [147…   df["deal_status"].value_counts()
```

Out[147…

```
deal_status
Offer Withdrawn     881
Funded              875
Unknown             822
No Deal             821
Rejected            820
Partially Funded    781
Name: count, dtype: int64
```

In [149…

```
df["investor"].value_counts()
```

Out[149…

```
investor
Barbara    490
Lori       478
Ashneer    463
Mark       459
Unknown    454
Namita     453
Aman       449
Peyush     445
Vineeta    444
Kevin      433
Anupam     432
Name: count, dtype: int64
```

Plots

UNIVARIATE PLOTS FOR CONTINOUS VARIABLE

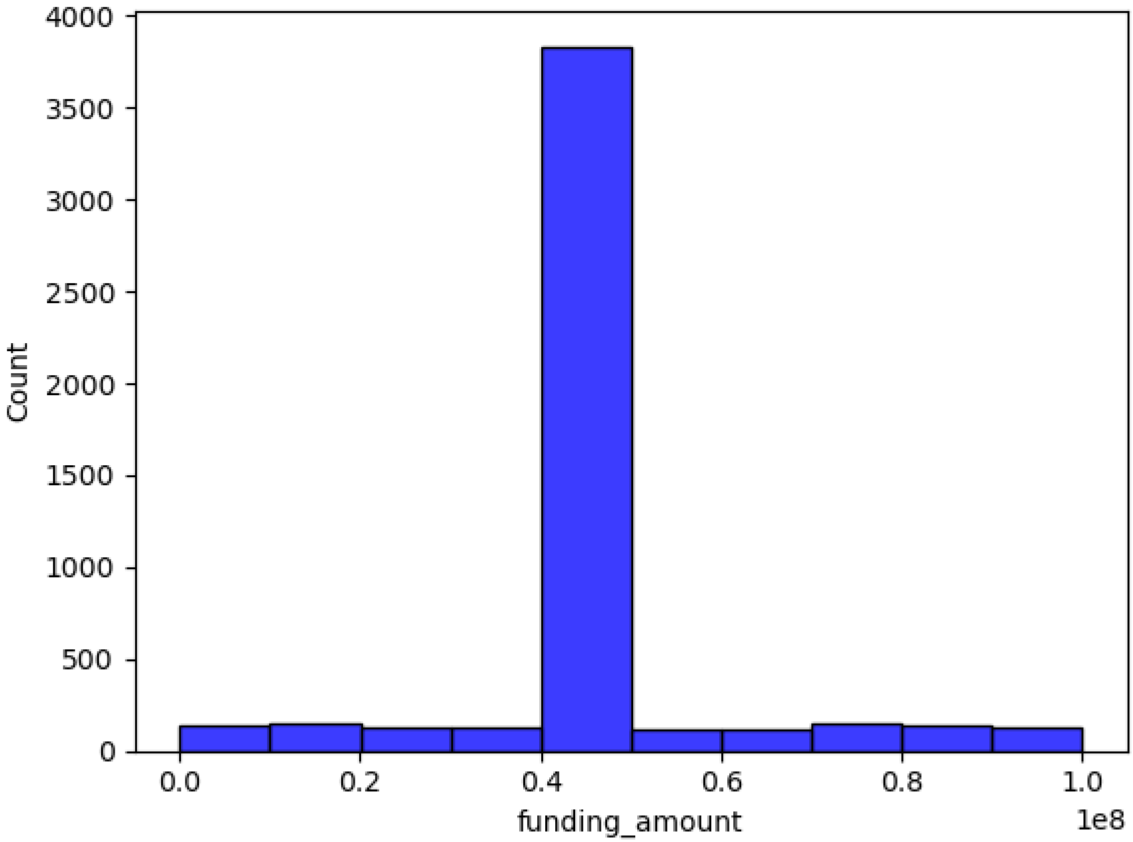
- HISTOGRAM
- KDE PLOT
- BOX PLOT

Histogram

Used to create and display a histogram using the Seaborn and Matplotlib libraries

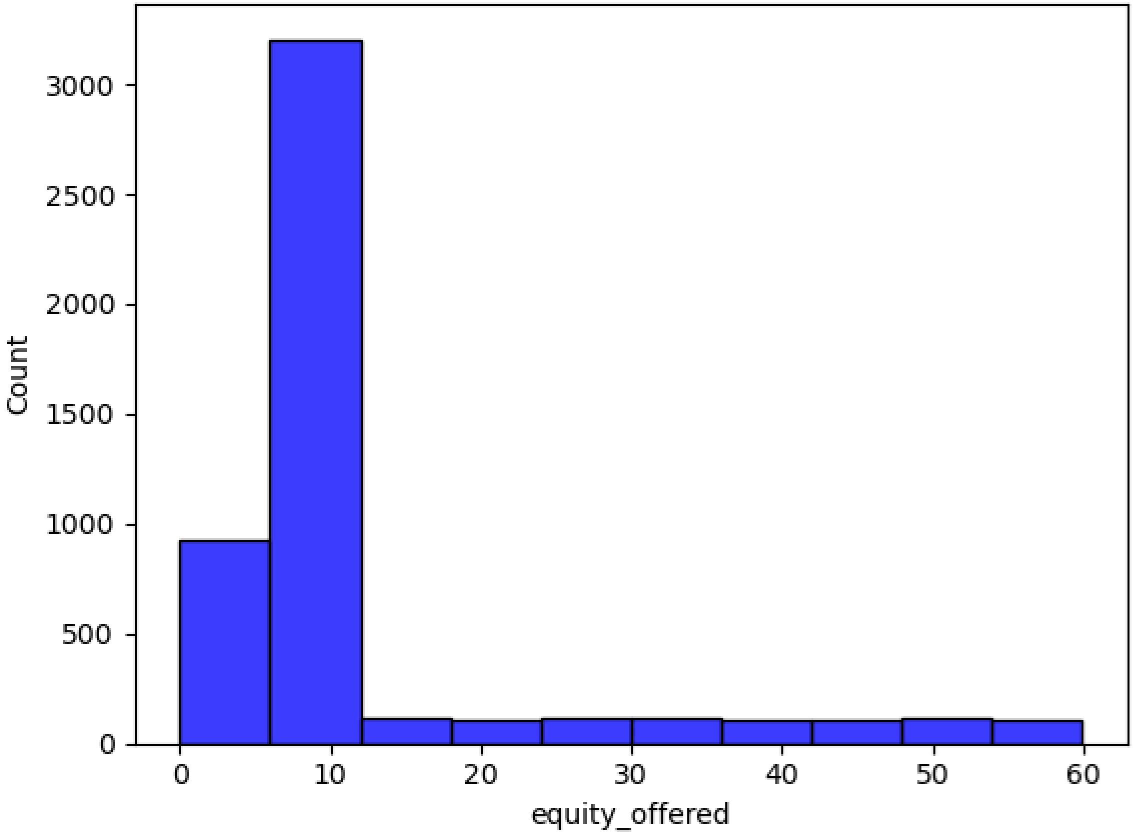
In [153…

```
# Create histogram
sns.histplot(df["funding_amount"], bins=10, color="blue", edgecolor="black")

plt.show()
```



In [154…

```
# Create histogram
sns.histplot(df["equity_offered"], bins=10, color="blue", edgecolor="black")

plt.show()
```

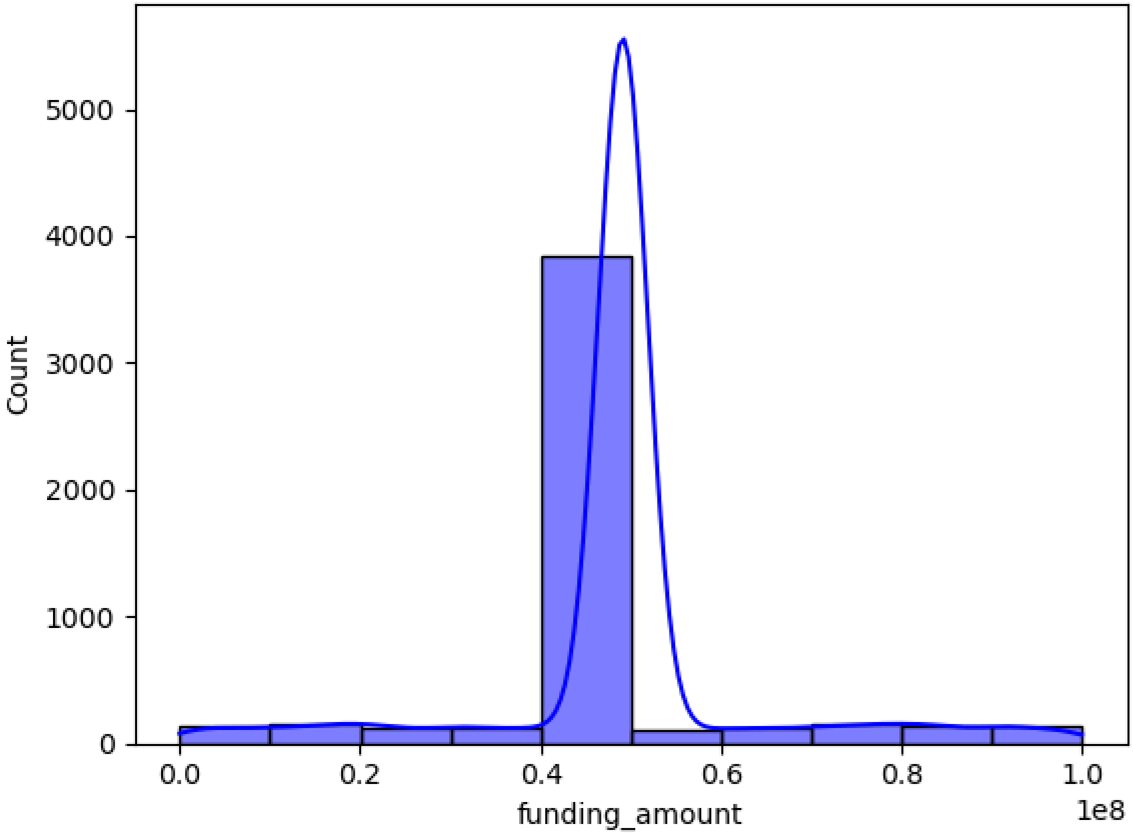


Kde plot

visualizes the probability density of a continuous variable

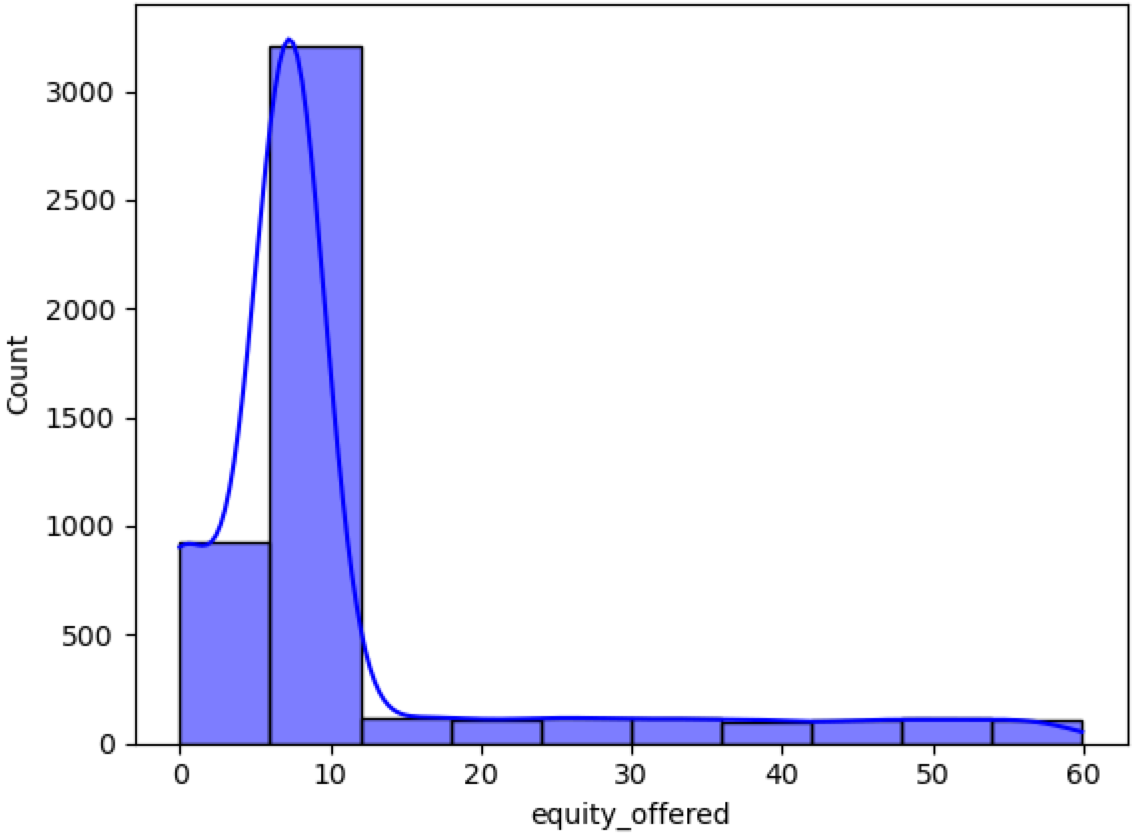
In [158…

```
sns.histplot(df["funding_amount"],bins = 10,color="blue", edgecolor="black",kde = T
plt.show()
```



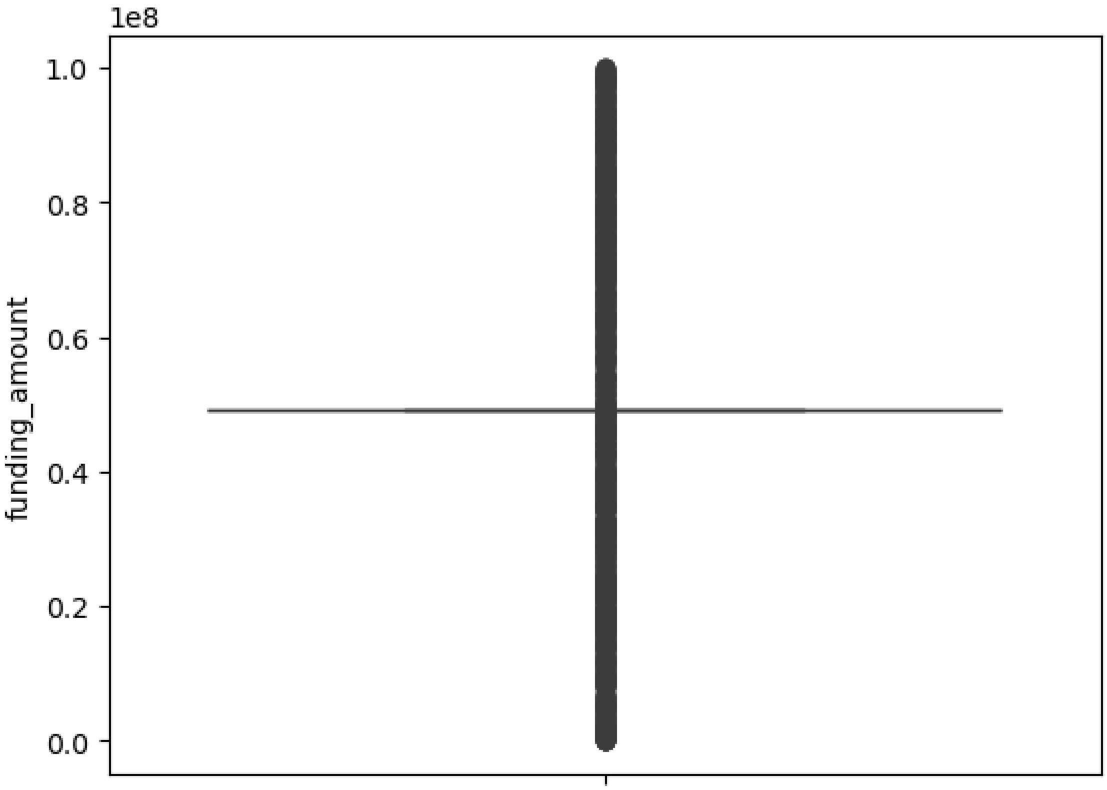
In [159…

```
sns.histplot(df["equity_offered"],bins = 10,color="blue", edgecolor="black",kde = T
plt.show()
```



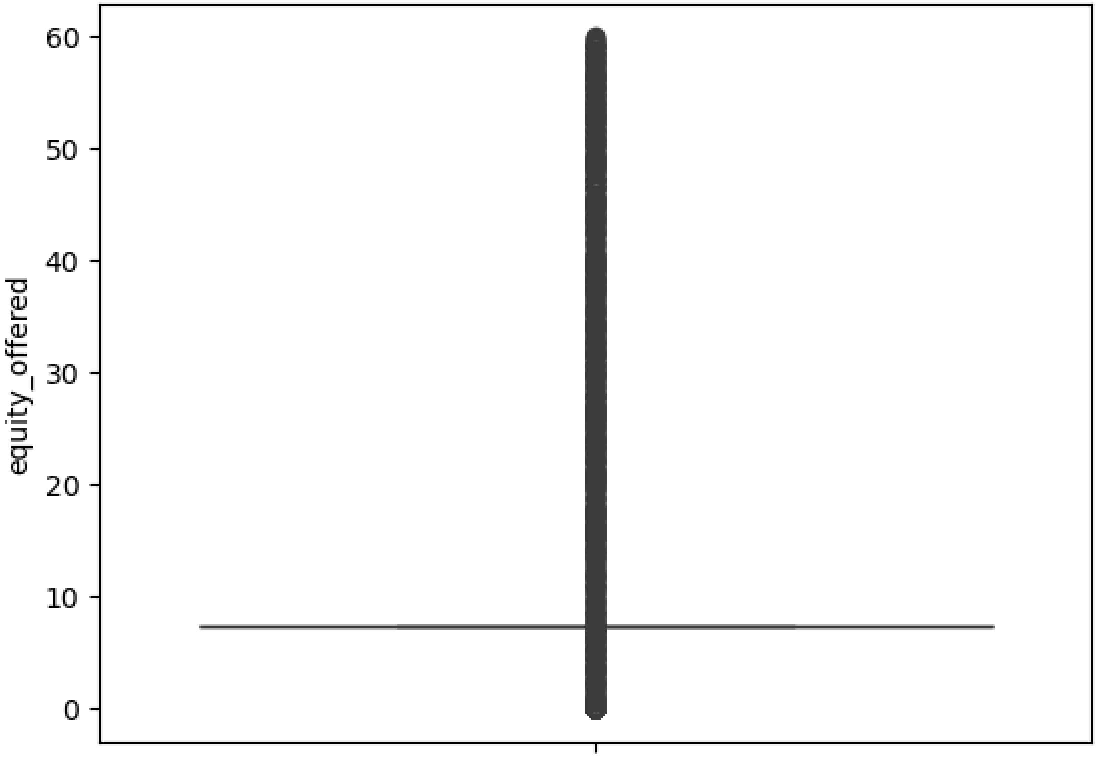
In [161…

```
sns.boxplot(df["funding_amount"])
plt.show()
```



In [163…

```
sns.boxplot(df["equity_offered"])
plt.show()
```



UNIVARIATE PLOTS FOR DISCRETE VARIABLE

- COUNT PLOT
- PIE CHART

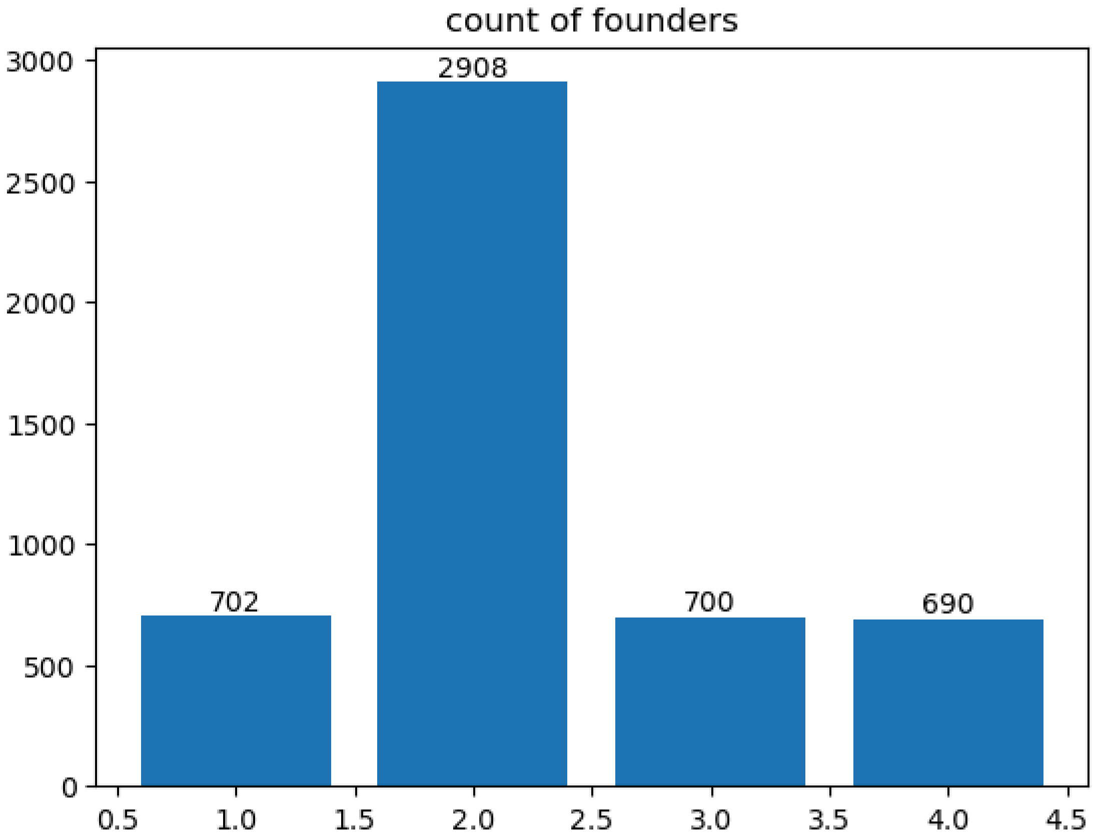
Count Plot

- Use: It shows the count of each category

visually represents the counts or frequencies of observations within different categories of a categorical variable

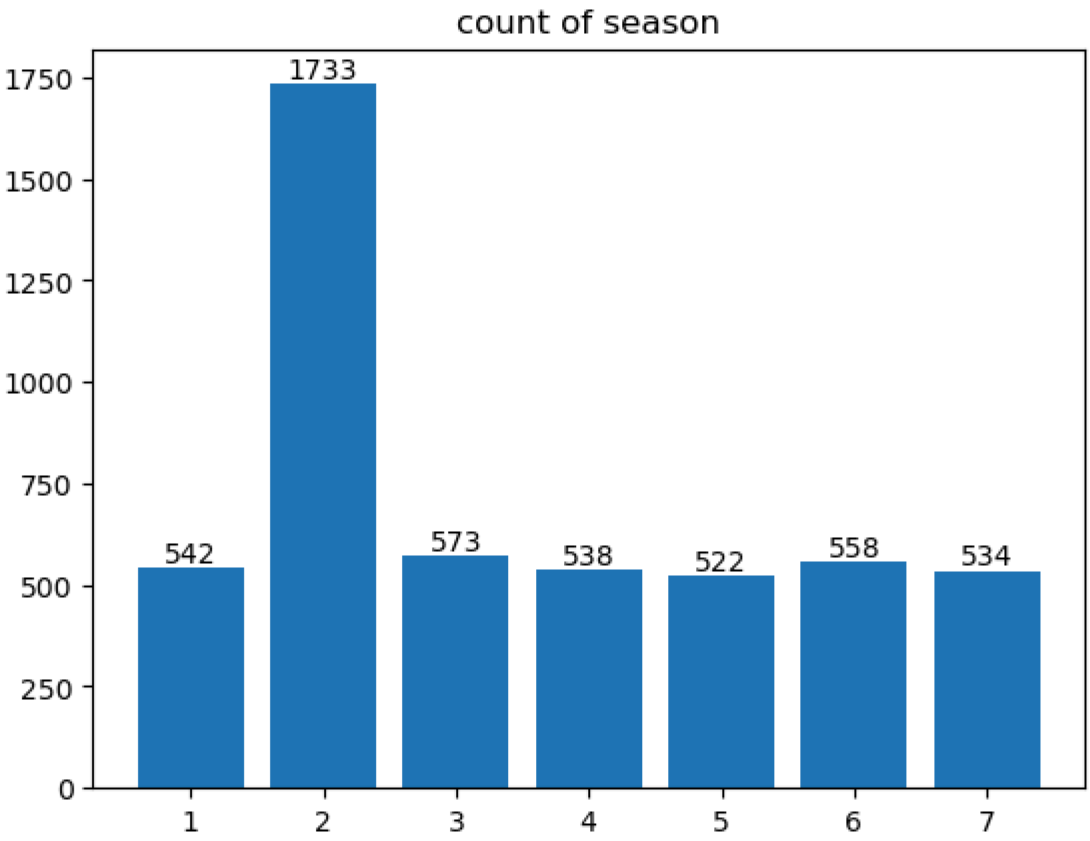
In [168…

```
patches = plt.bar(df["founders"].value_counts().index,df["founders"].value_counts()
plt.bar_label(patches)
plt.title("count of founders")
plt.show()
```



In [169…

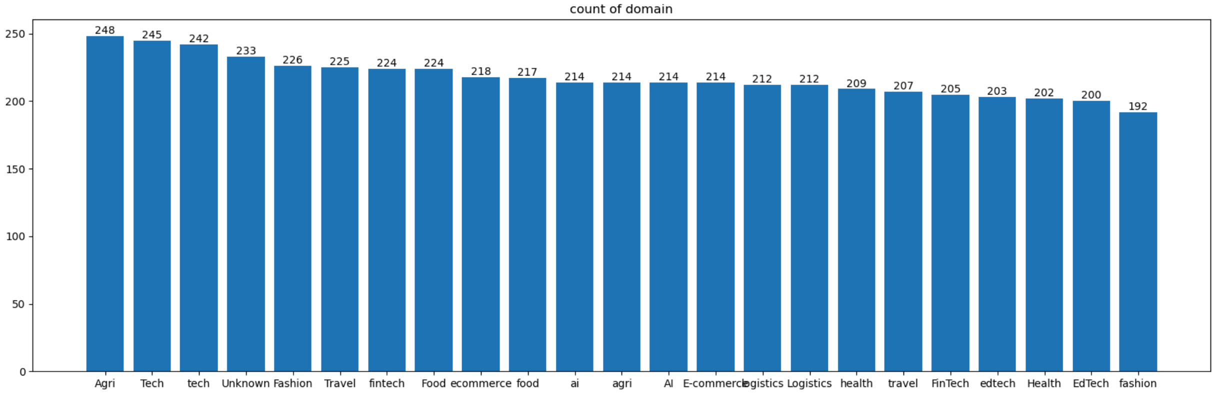
```
patches = plt.bar(df["season"].value_counts().index,df["season"].value_counts())
plt.bar_label(patches)
plt.title("count of season")
plt.show()
```



Categorical

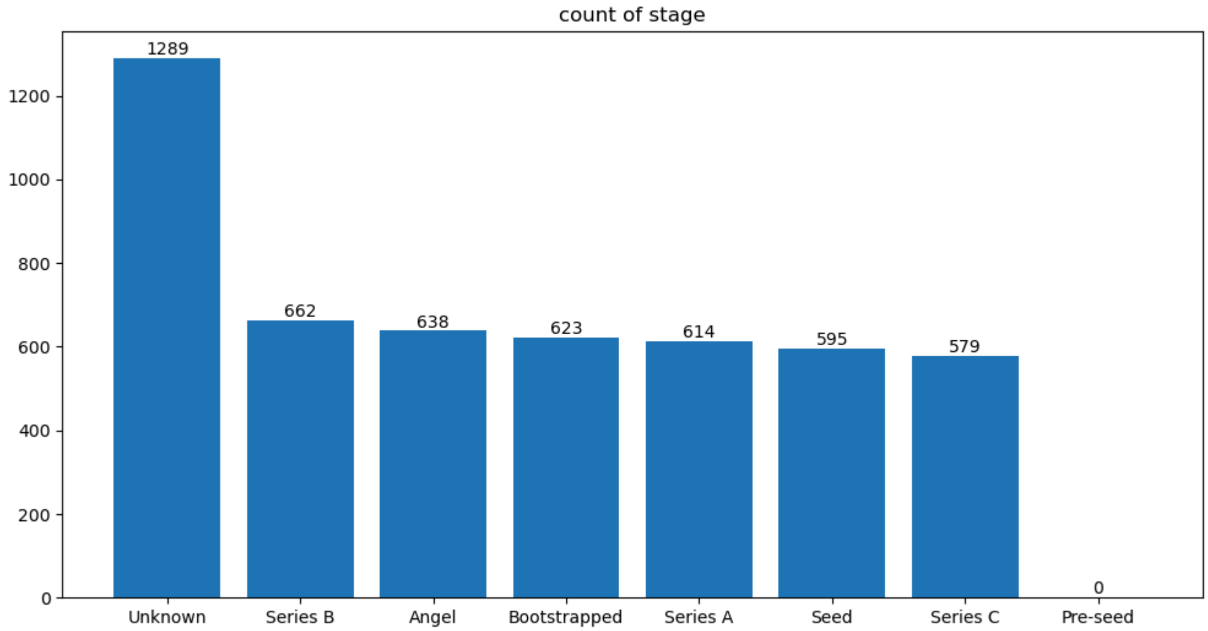
In [173…

```
plt.figure(figsize=(20, 6))

patches = plt.bar(df["domain"].value_counts().index,df["domain"].value_counts())
plt.bar_label(patches)
plt.title("count of domain")
plt.show()
```



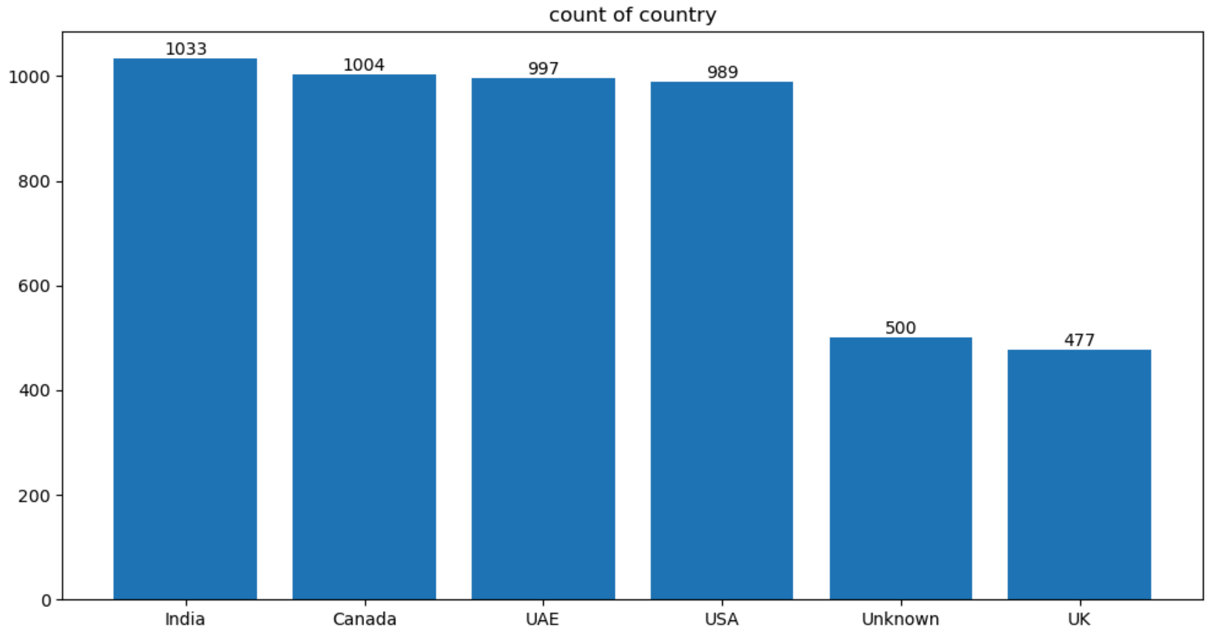
In [174…

```
plt.figure(figsize=(12, 6))

patches = plt.bar(df["stage"].value_counts().index,df["stage"].value_counts())
plt.bar_label(patches)
plt.title("count of stage")
plt.show()
```



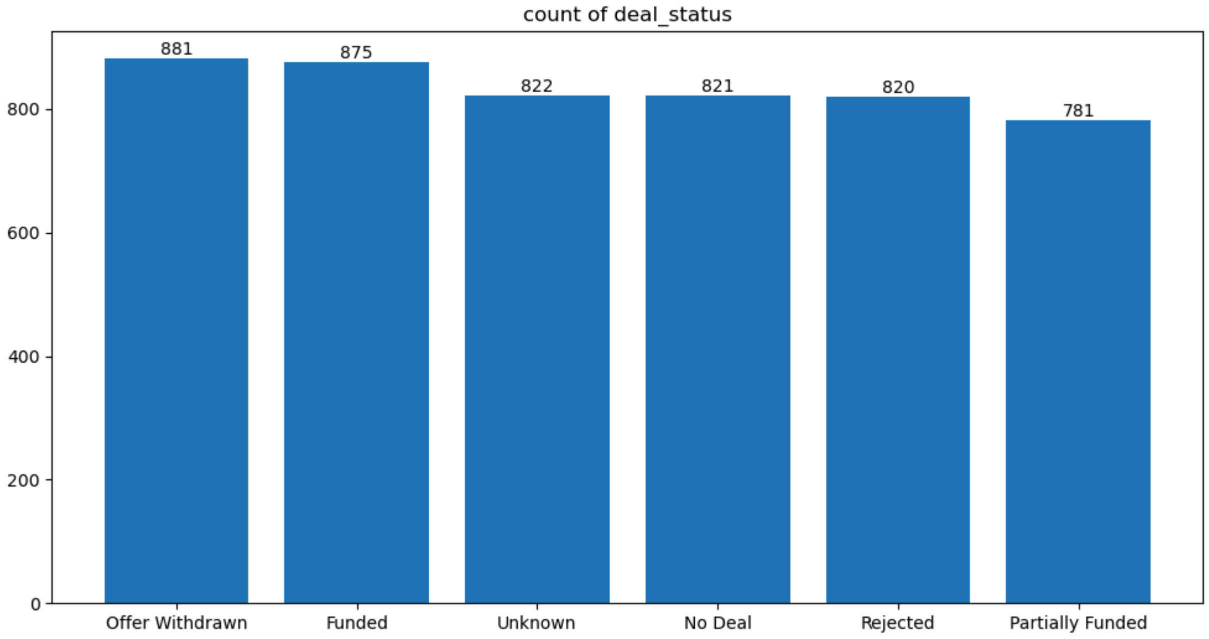
In [175…

```
plt.figure(figsize=(12, 6))

patches = plt.bar(df["country"].value_counts().index,df["country"].value_counts())
plt.bar_label(patches)
plt.title("count of country")
plt.show()
```



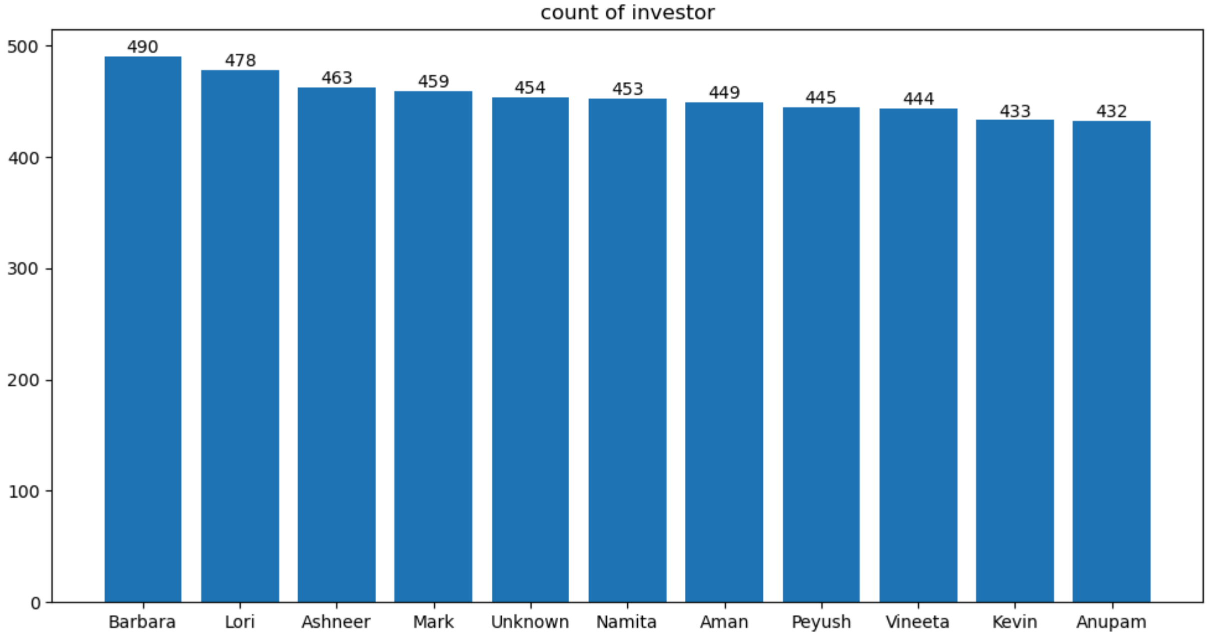
In [177…

```
plt.figure(figsize=(12, 6))

patches = plt.bar(df["deal_status"].value_counts().index,df["deal_status"].value_co
plt.bar_label(patches)
plt.title("count of deal_status")
plt.show()
```



In [179…

```
plt.figure(figsize=(12, 6))

patches = plt.bar(df["investor"].value_counts().index,df["investor"].value_counts()
plt.bar_label(patches)
plt.title("count of investor")
plt.show()
```

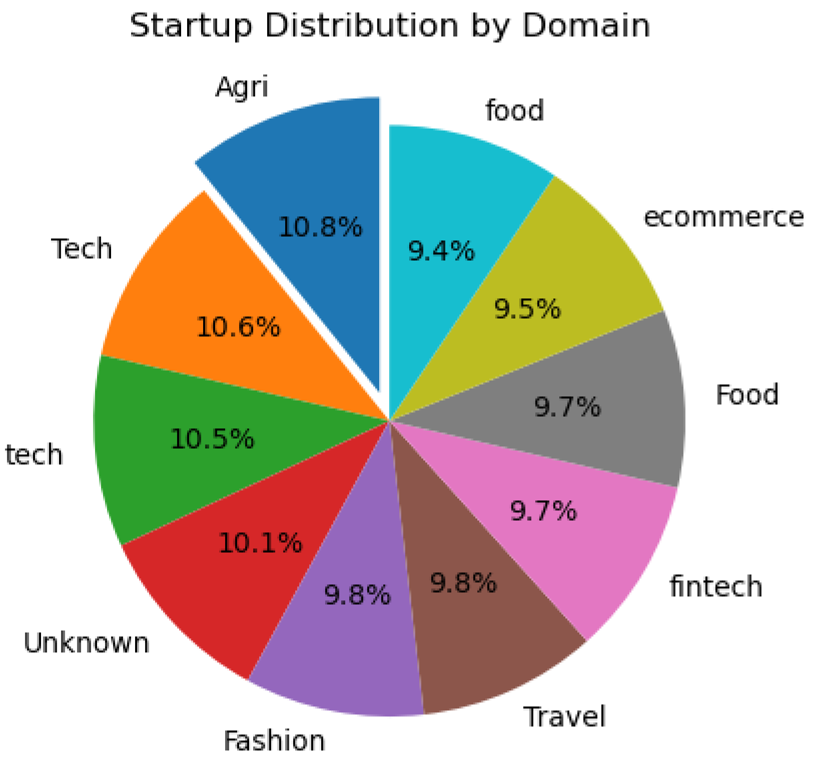


Pie Plot

is a circular statistical graphic that visualizes data proportions

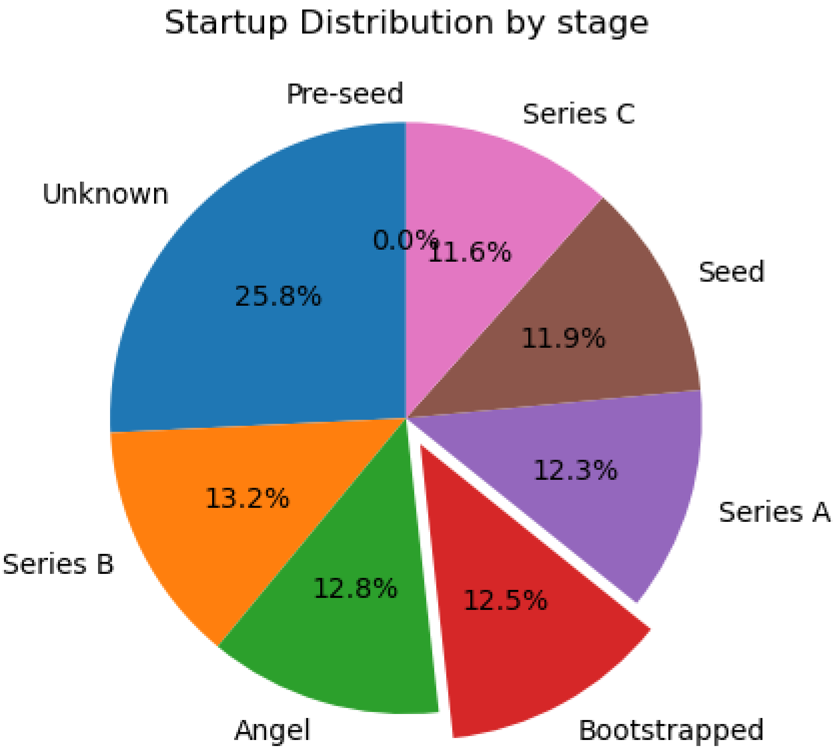
In [183…

```
d = df["domain"].value_counts().head(10)

plt.pie(d, labels=d.index, autopct="%0.1f%%", startangle=90, explode=[0.1]+[0]*9)
plt.title("Startup Distribution by Domain")
plt.show()
```



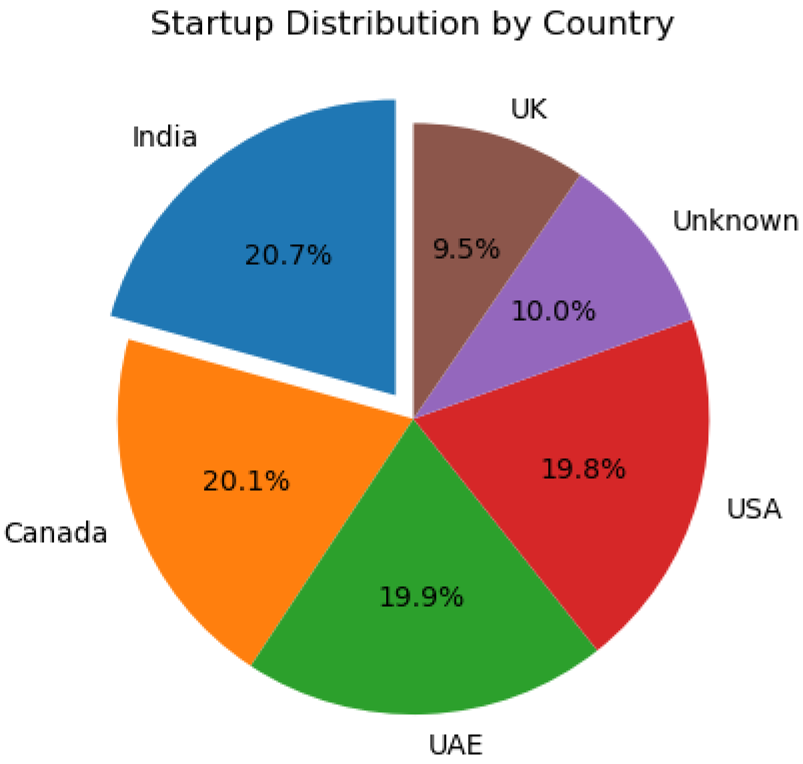
In [185…

```
d = df["stage"].value_counts()

plt.pie(d, labels=d.index, autopct="%0.1f%%", startangle=90, explode=[0,0,0,0.1,0,0
plt.title("Startup Distribution by stage")
plt.show()
```



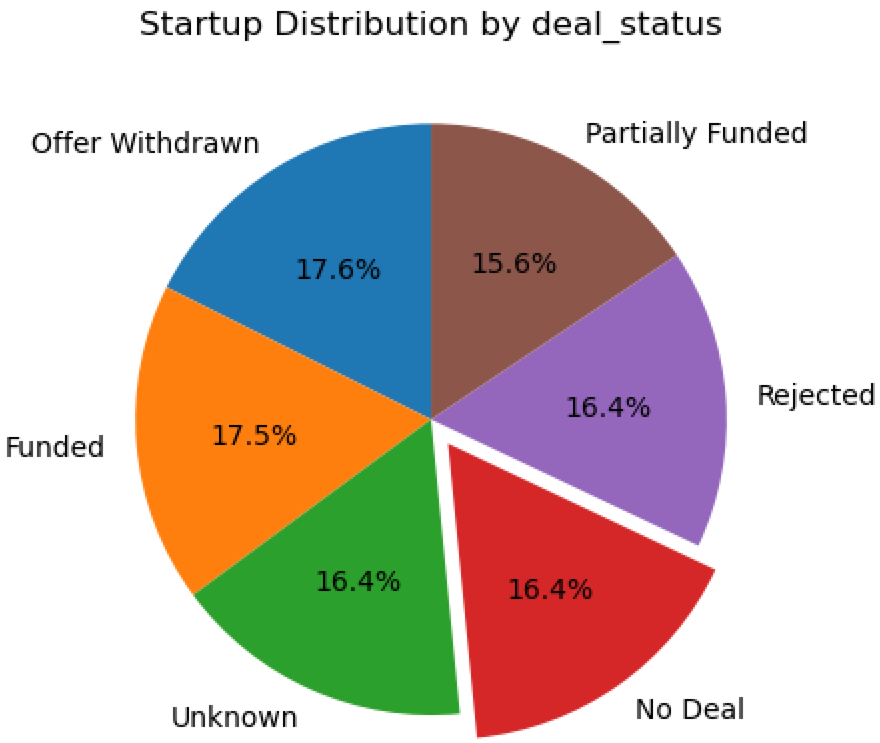
In [188…

```
d = df["country"].value_counts()

plt.pie(d, labels=d.index, autopct="%0.1f%%", startangle=90, explode=[0.1]+[0]*5)
plt.title("Startup Distribution by Country")
plt.show()
```



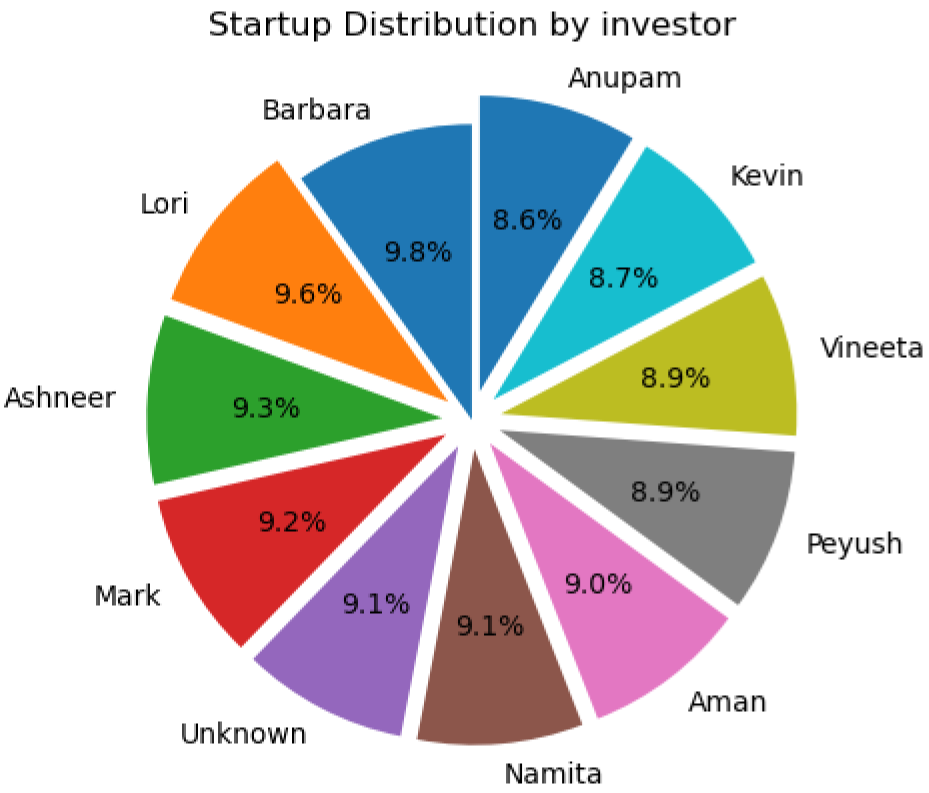
In [190…

```
d = df["deal_status"].value_counts()

plt.pie(d, labels=d.index, autopct="%0.1f%%", startangle=90, explode=[0,0,0,0.1,0,0
plt.title("Startup Distribution by deal_status")
plt.show()
```



In [192…

```
d = df["investor"].value_counts()

plt.pie(d, labels=d.index, autopct="%0.1f%%", startangle=90, explode=[0]+[0.1]*10)
plt.title("Startup Distribution by investor")
plt.show()
```



Bivariate Measures

- crosstab (2 discrete variable)
- correlation (Continuous + Continuous)
- groupby (1 discrete + 1 continous)

Correlation

- corr() is used to find the pairwise correlation of all columns in the Pandas Dataframe in Python

In [196…

```
df[continous].corr()
```

Out[196…

	funding_amount	equity_offered
funding_amount	1	-0
equity_offered	-0	1

CrossTab

- Cosstab returns the contingency table resulting from crossing two or more fields in a dataframe

In [199…

```
pd.crosstab(df["investor"], df["domain"], margins=True)
```

Out[199…

domain	AI	Agri	EdTech	E-commerce	Fashion	FinTech	Food	Health	Logistics	Tec
investor										
Aman	15	19	16	11	21	22	17	13	28	1
Anupam	17	27	14	27	18	17	11	13	15	2
Ashneer	17	22	17	21	22	21	22	18	18	2
Barbara	17	25	25	19	29	19	28	30	26	2
Kevin	18	22	19	20	19	14	25	22	17	2
Lori	20	28	20	20	20	15	24	21	15	1
Mark	27	22	16	18	16	22	28	23	21	2
Namita	24	16	23	24	20	26	19	15	18	2
Peyush	21	19	19	18	23	14	16	16	18	2
Unknown	20	32	18	18	18	21	21	13	15	2
Vineeta	18	16	13	18	20	14	13	18	21	1
All	214	248	200	214	226	205	224	202	212	24

12 rows × 24 columns

In [201…

```
pd.crosstab(df["investor"], df["stage"], margins=True)
```

Out[201…

stage	Bootstrapped	Seed	Angel	Series A	Series B	Series C	Unknown	All
investor								
Aman	49	40	67	63	60	53	117	449
Anupam	54	55	56	54	55	53	105	432
Ashneer	63	61	61	56	57	48	117	463
Barbara	64	53	64	66	64	51	128	490
Kevin	49	57	50	50	68	48	111	433
Lori	58	65	48	58	57	64	128	478
Mark	61	52	58	60	70	54	104	459
Namita	54	54	46	49	63	52	135	453
Peyush	56	62	58	51	56	50	112	445
Unknown	54	47	60	59	49	65	120	454
Vineeta	61	49	70	48	63	41	112	444
All	623	595	638	614	662	579	1289	5000

In [203…

```
pd.crosstab(df["investor"], df["country"], margins=True)
```

Out[203…

country	Canada	India	UAE	UK	USA	Unknown	All
investor							
Aman	92	86	89	41	84	57	449
Anupam	90	98	94	34	83	33	432
Ashneer	90	102	95	48	84	44	463
Barbara	100	111	93	40	106	40	490
Kevin	95	83	93	41	85	36	433
Lori	93	93	90	56	92	54	478
Mark	83	97	84	39	104	52	459
Namita	108	87	97	30	82	49	453
Peyush	82	82	98	43	87	53	445
Unknown	93	95	80	57	87	42	454
Vineeta	78	99	84	48	95	40	444
All	1004	1033	997	477	989	500	5000

In [205…

```
pd.crosstab(df["investor"], df["deal_status"], margins=True)
```

Out[205…

deal_status	Funded	No Deal	Offer Withdrawn	Partially Funded	Rejected	Unknown	All
investor							
Aman	76	75	73	75	79	71	449
Anupam	74	67	76	67	80	68	432
Ashneer	88	71	89	72	74	69	463
Barbara	88	80	90	77	65	90	490
Kevin	77	82	66	57	82	69	433
Lori	72	79	81	81	71	94	478
Mark	84	79	78	71	75	72	459
Namita	79	77	84	66	81	66	453
Peyush	74	71	85	59	74	82	445
Unknown	80	72	82	76	72	72	454
Vineeta	83	68	77	80	67	69	444
All	875	821	881	781	820	822	5000

If you want to see proportions (percentages):

In [208…

```
pd.crosstab(df["investor"], df["deal_status"], normalize="index") * 100
```

Out[208…

deal_status	Funded	No Deal	Offer Withdrawn	Partially Funded	Rejected	Unknown
investor						
Aman	17	17	16	17	18	16
Anupam	17	16	18	16	19	16
Ashneer	19	15	19	16	16	15
Barbara	18	16	18	16	13	18
Kevin	18	19	15	13	19	16
Lori	15	17	17	17	15	20
Mark	18	17	17	15	16	16
Namita	17	17	19	15	18	15
Peyush	17	16	19	13	17	18
Unknown	18	16	18	17	16	16
Vineeta	19	15	17	18	15	16

Group By

- Groupby separate identical data into groups to allow for further aggregation and analysis

In [211…

```
df.groupby("startup_name")["funding_amount"].describe().T
```

Out[211…

startup_name	AIBox_1247	AIBox_1825	AIBox_1855	AIBox_2035	AIBox_2054	AIBox_2259
count	1	1	1	1	1	1
mean	94,477,539	49,114,685	32,617,072	49,114,685	49,114,685	49,114,685
std	NaN	NaN	NaN	NaN	NaN	NaN
min	94,477,539	49,114,685	32,617,072	49,114,685	49,114,685	49,114,685
25%	94,477,539	49,114,685	32,617,072	49,114,685	49,114,685	49,114,685
50%	94,477,539	49,114,685	32,617,072	49,114,685	49,114,685	49,114,685
75%	94,477,539	49,114,685	32,617,072	49,114,685	49,114,685	49,114,685
max	94,477,539	49,114,685	32,617,072	49,114,685	49,114,685	49,114,685

8 rows × 5000 columns

In [212…

```
df.groupby("domain")["funding_amount"].describe().T
```

Out[212…

domain	AI	Agri	EdTech	E-commerce	Fashion	FinTech	Food
count	214	248	200	214	226	205	224
mean	47,249,266	48,942,867	49,620,698	49,498,481	48,685,602	45,856,652	49,256,378
std	14,615,753	14,764,323	13,718,598	13,969,906	13,290,577	14,840,527	13,343,737
min	1,032,192	203,969	394,038	106,075	4,147,173	285,251	128,510
25%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
50%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
75%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
max	99,223,531	98,169,786	97,772,241	98,845,729	98,669,891	99,948,211	94,129,469

8 rows × 23 columns

In [213…

```
df.groupby("stage")["funding_amount"].describe().T
```

```
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_19244\4048560205.py:1: FutureWarning:
The default of observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or observed=True t
o adopt the future default and silence this warning.
  df.groupby("stage")["funding_amount"].describe().T
```

Out[213…

stage	Bootstrapped	Pre-seed	Seed	Angel	Series A	Series B	Series C	Unkr
count	623	0	595	638	614	662	579	[illegible]
mean	50,129,787	NaN	47,788,696	48,774,616	49,343,368	48,970,393	49,900,248	49,79
std	14,657,350	NaN	15,665,005	13,881,888	15,351,101	15,422,852	14,183,466	15,03
min	203,969	NaN	106,075	19,089	29,030	177,420	128,510	28
25%	49,114,685	NaN	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,11
50%	49,114,685	NaN	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,11
75%	49,114,685	NaN	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,11
max	99,674,401	NaN	98,829,135	99,707,818	99,698,789	99,965,336	99,948,211	99,78

In [214…

```
df.groupby("country")["funding_amount"].describe().T
```

Out[214…

country	Canada	India	UAE	UK	USA	Unknown
count	1,004	1,033	997	477	989	500
mean	49,353,495	49,391,704	48,867,411	50,389,367	49,382,048	48,827,548
std	14,366,021	14,944,605	14,335,554	15,873,133	15,500,549	15,090,325
min	498,126	19,089	106,075	203,969	128,510	223,111
25%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
50%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
75%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
max	99,948,211	99,707,818	99,965,336	99,950,959	99,369,892	99,223,531

In [215…

```
df.groupby("deal_status")["funding_amount"].describe().T
```

Out[215…

deal_status	Funded	No Deal	Offer Withdrawn	Partially Funded	Rejected	Unknown
count	875	821	881	781	820	822
mean	49,712,416	48,484,638	48,888,635	49,863,201	49,502,095	49,478,926
std	14,161,802	15,191,389	14,486,363	14,978,527	15,493,605	15,294,757
min	177,420	128,510	106,075	29,030	285,251	19,089
25%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
50%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
75%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
max	99,965,336	98,383,186	99,950,959	99,369,892	99,474,284	99,707,818

In [216…

```
df.groupby("investor")["funding_amount"].describe().T
```

Out[216…

investor	Aman	Anupam	Ashneer	Barbara	Kevin	Lori	Mark	
count	449	432	463	490	433	478	459	
mean	48,658,004	49,134,542	48,641,630	49,045,526	50,018,240	48,694,201	48,940,495	4
std	16,228,795	15,609,212	15,102,020	15,137,877	13,395,866	14,122,156	14,954,017	1
min	813,173	394,038	285,251	203,969	2,315,681	177,420	106,075	
25%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	4
50%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	4
75%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	4
max	99,369,892	99,707,818	98,190,264	99,674,401	99,965,336	99,948,211	98,115,705	9

In [217…

```
df.groupby("startup_name")["equity_offered"].describe().T
```

Out[217…

startup_name	AIBox_1247	AIBox_1825	AIBox_1855	AIBox_2035	AIBox_2054	AIBox_2259
count	1	1	1	1	1	1
mean	7	7	7	7	7	24
std	NaN	NaN	NaN	NaN	NaN	NaN
min	7	7	7	7	7	24
25%	7	7	7	7	7	24
50%	7	7	7	7	7	24
75%	7	7	7	7	7	24
max	7	7	7	7	7	24

8 rows × 5000 columns

In [218…

```
df.groupby("domain")["equity_offered"].describe().T
```

Out[218…

domain	AI	Agri	EdTech	E-commerce	Fashion	FinTech	Food	Health	Logistics	Tech
count	214	248	200	214	226	205	224	202	212	245
mean	11	10	11	12	10	12	11	10	12	12
std	12	11	13	15	11	14	13	11	14	15
min	0	0	0	0	0	0	0	0	0	0
25%	7	7	7	7	7	7	7	7	7	7
50%	7	7	7	7	7	7	7	7	7	7
75%	7	7	7	7	7	7	7	7	7	7
max	59	57	59	58	58	57	57	54	59	60

8 rows × 23 columns

In [219…

```
df.groupby("stage")["equity_offered"].describe().T
```

```
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_19244\2048886722.py:1: FutureWarning:
The default of observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or observed=True t
o adopt the future default and silence this warning.
  df.groupby("stage")["equity_offered"].describe().T
```

Out[219…

stage	Bootstrapped	Pre-seed	Seed	Angel	Series A	Series B	Series C	Unknown
count	623	0	595	638	614	662	579	1,289
mean	11	NaN	11	11	11	11	11	11
std	13	NaN	13	13	14	13	12	13
min	0	NaN	0	0	0	0	0	0
25%	7	NaN	7	7	7	7	7	7
50%	7	NaN	7	7	7	7	7	7
75%	7	NaN	7	7	7	7	7	7
max	59	NaN	60	60	59	59	59	60

In [220…

```
df.groupby("country")["equity_offered"].describe().T
```

Out[220…

country	Canada	India	UAE	UK	USA	Unknown
count	1,004	1,033	997	477	989	500
mean	11	11	11	11	11	11
std	13	13	14	13	13	14
min	0	0	0	0	0	0
25%	7	7	7	7	7	7
50%	7	7	7	7	7	7
75%	7	7	7	7	7	7
max	59	60	59	60	60	59

In [221…

```
df.groupby("deal_status")["equity_offered"].describe().T
```

Out[221…

deal_status	Funded	No Deal	Offer Withdrawn	Partially Funded	Rejected	Unknown
count	875	821	881	781	820	822
mean	13	12	14	13	0	13
std	13	13	14	13	0	13
min	1	1	1	1	0	1
25%	7	7	7	7	0	7
50%	7	7	7	7	0	7
75%	7	7	7	7	0	7
max	59	60	60	60	0	60

In [222…

```
df.groupby("investor")["equity_offered"].describe().T
```

Out[222…

investor	Aman	Anupam	Ashneer	Barbara	Kevin	Lori	Mark	Namita	Peyush	Unknow
count	449	432	463	490	433	478	459	453	445	45
mean	11	11	12	11	10	12	12	11	10	1
std	13	13	14	13	12	14	14	13	11	1
min	0	0	0	0	0	0	0	0	0	
25%	7	7	7	7	7	7	7	7	7	
50%	7	7	7	7	7	7	7	7	7	
75%	7	7	7	7	7	7	7	7	7	
max	60	60	60	60	59	59	59	57	59	5

Bivariate Plots

- SCATTER PLOT
- BAR PLOT
- JOINT PLOT
- HEAT MAP

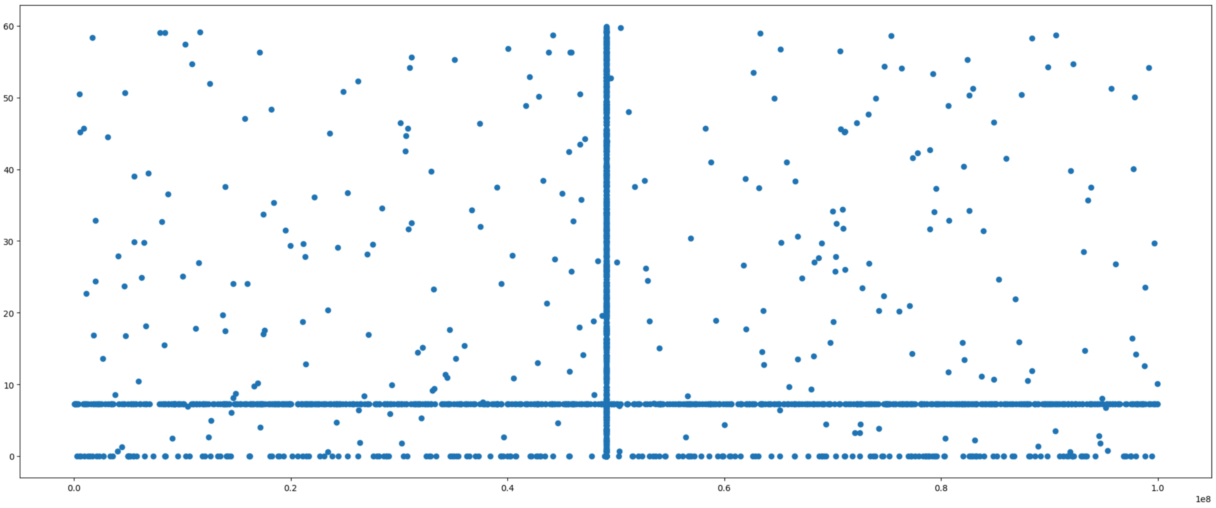
Scatter Plot

- A scatter plot in Python is a type of data visualization that uses dots to represent values for two different numeric variables
- Marking the data points on the graph
- 2 continous

USE: To check 1.linearity,2.Direction,3.Strength

In [225…

```
plt.figure(figsize=(25,10))

plt.scatter(x=df["funding_amount"],y=df["equity_offered"])
plt.show()
```



Joint Plot

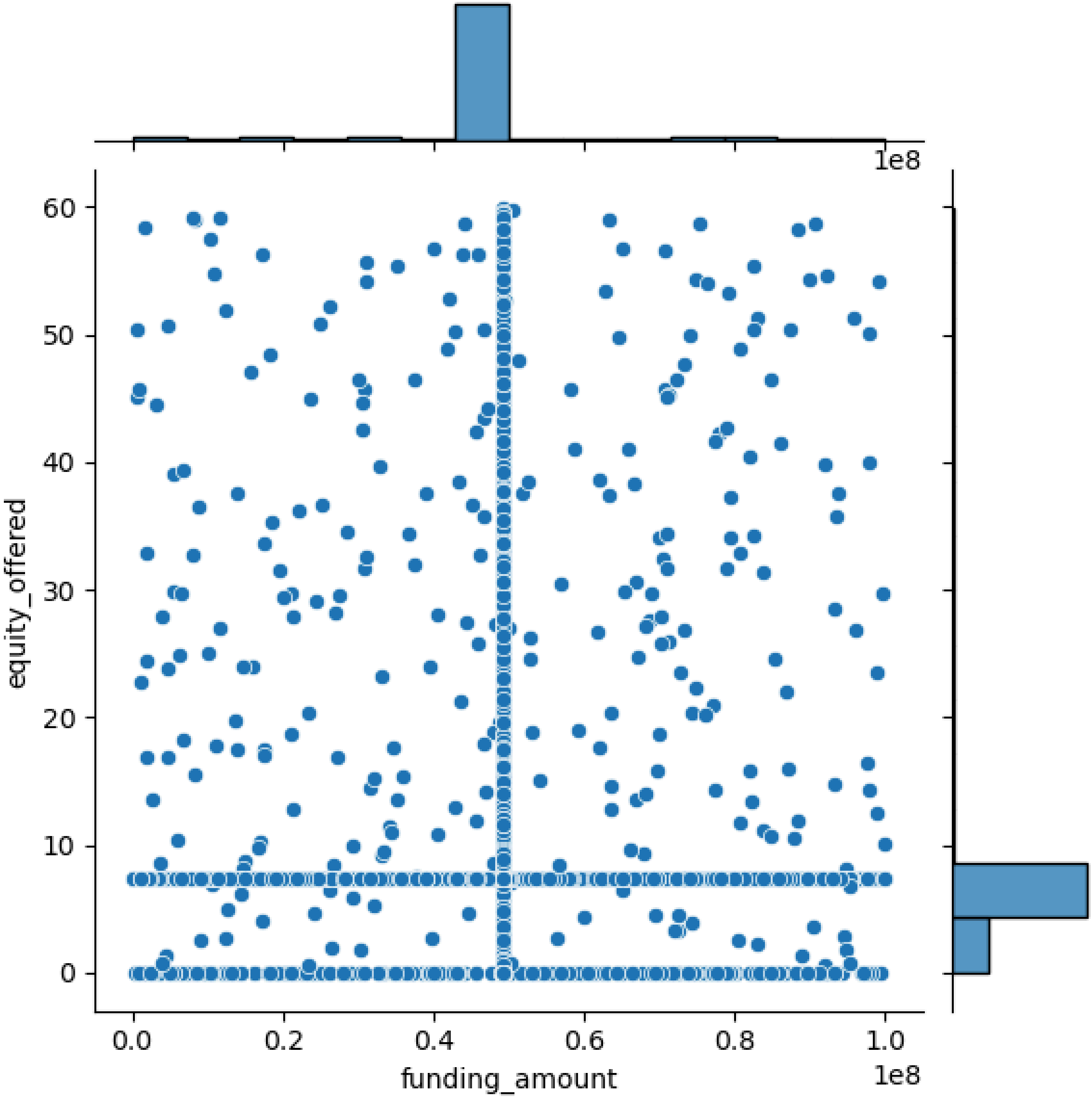
- joint plot visualizes the relationship between two variables along with their individual distributions

Use: Combination of Scatter plot and Histogram

- A join plot allows to study the relationship between 2 numeric variables.

In [230…

```
sns.jointplot(x="funding_amount",y="equity_offered",data=df)
plt.show()
```

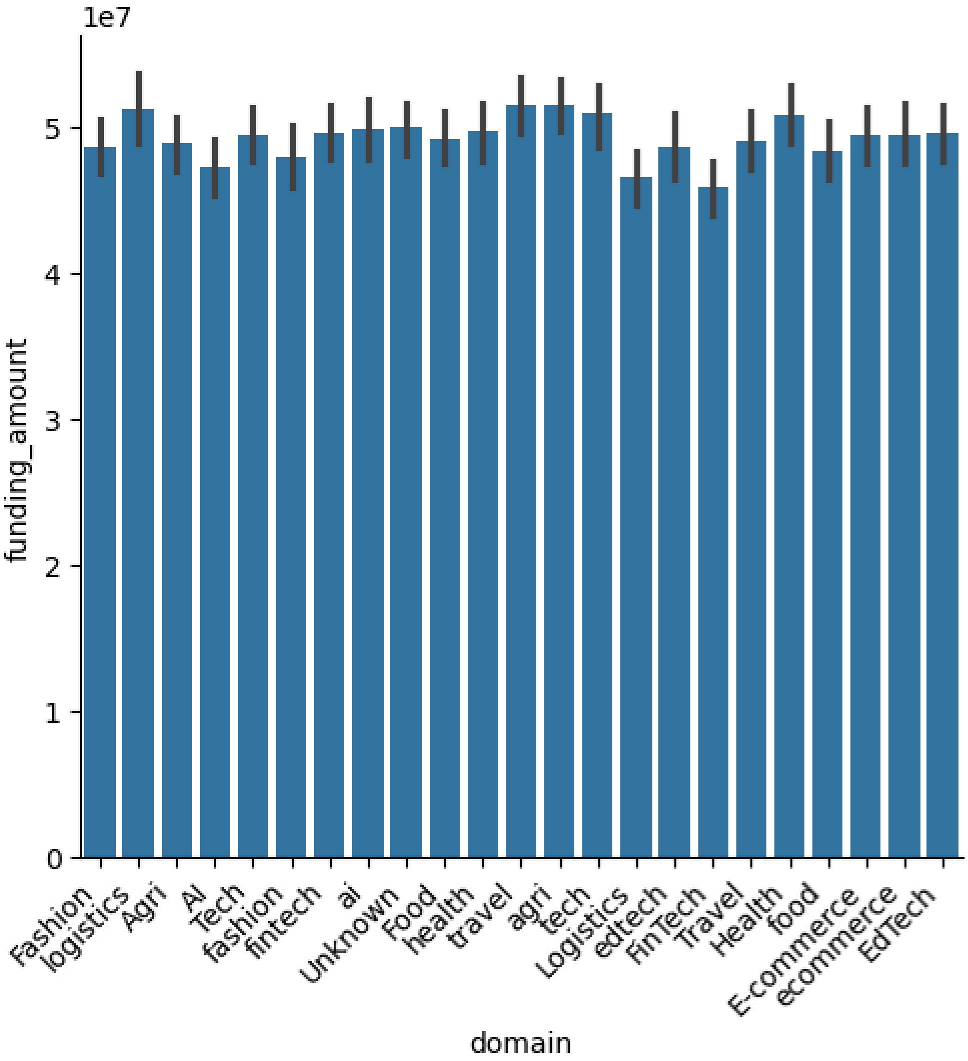


Bar plot

- is a data visualization tool used to represent categorical data with rectangular bars

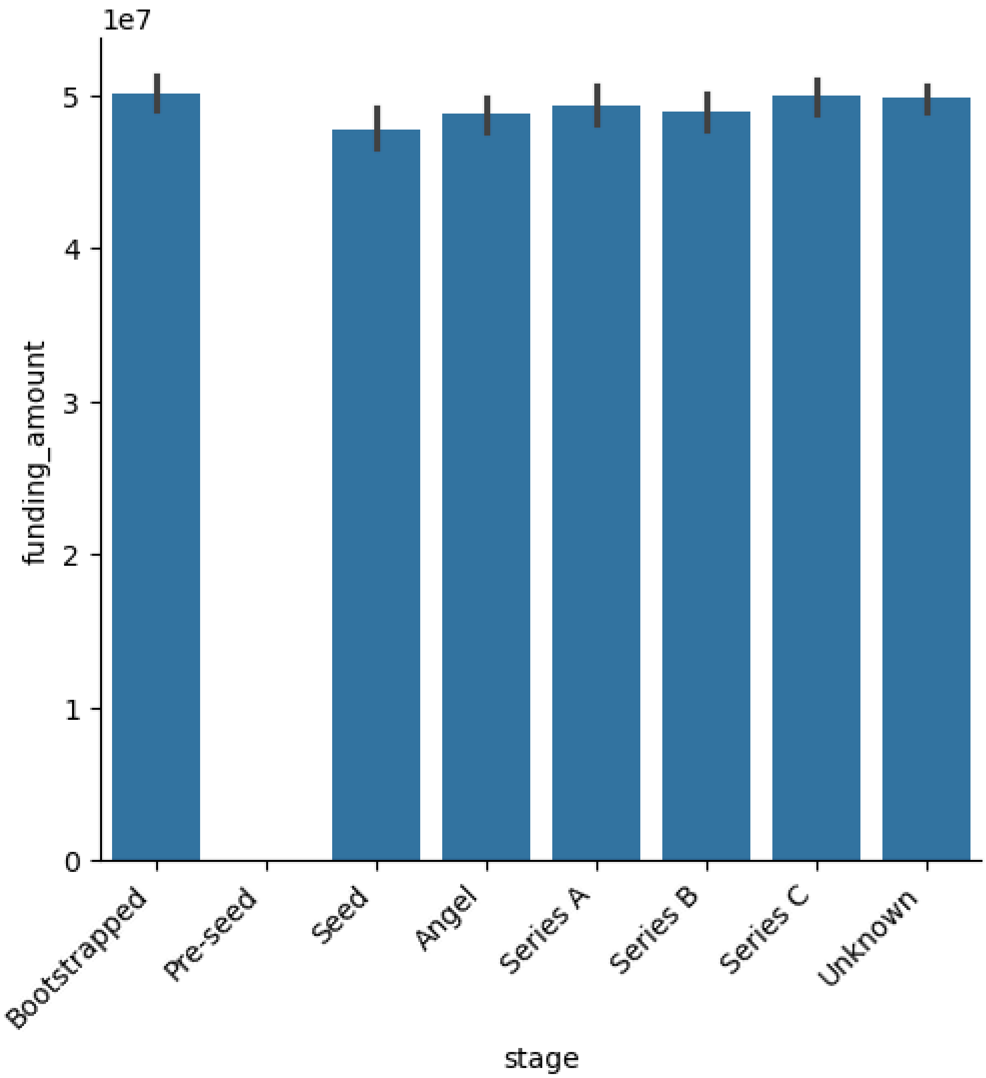
In [235…

```
sns.catplot(x="domain",y="funding_amount",data=df,kind="bar")
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for readability
plt.show()
```



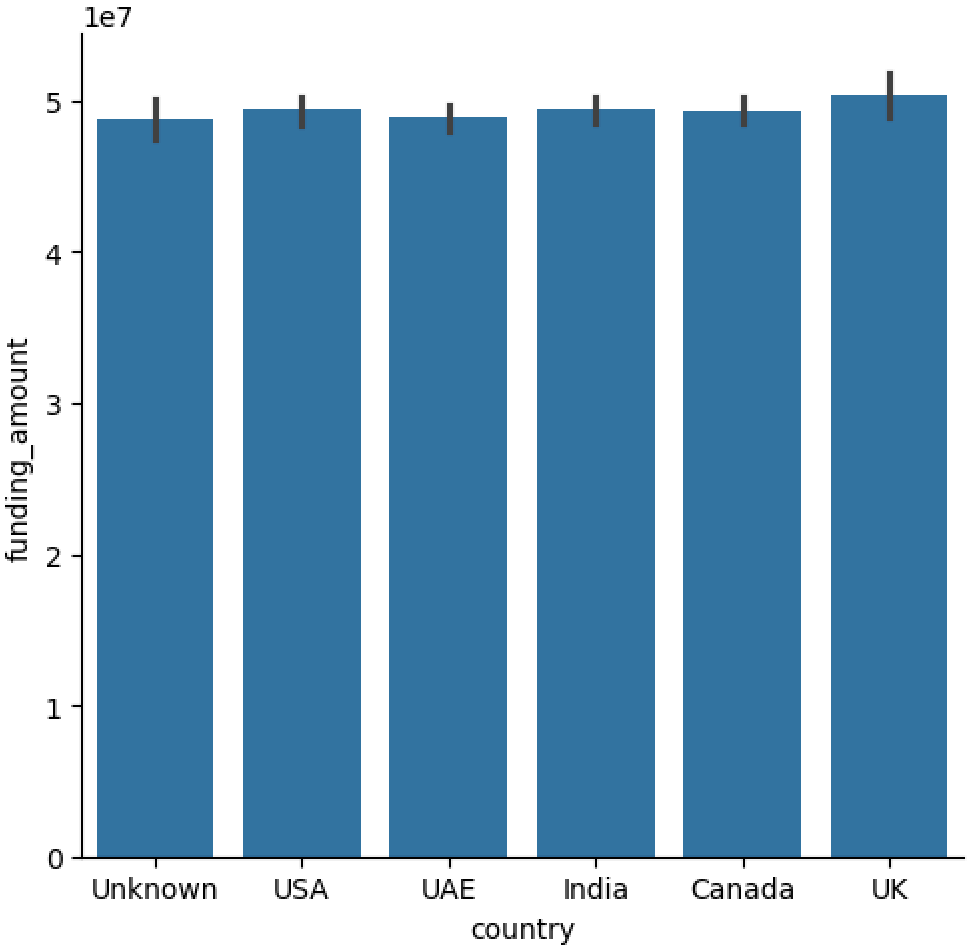
In [236…

```
sns.catplot(x="stage",y="funding_amount",data=df,kind="bar")
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for readability
plt.show()
```



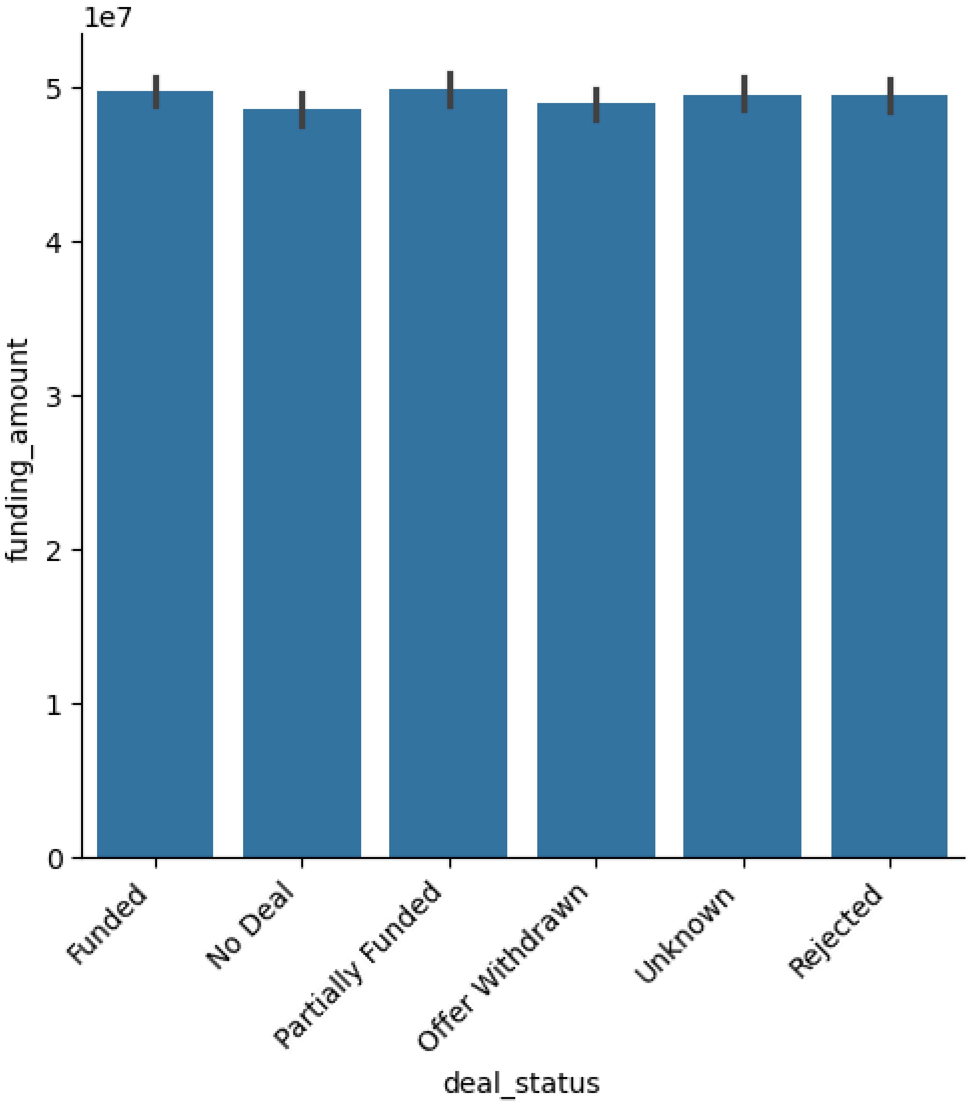
In [237…

```
sns.catplot(x="country",y="funding_amount",data=df,kind="bar")
plt.show()
```



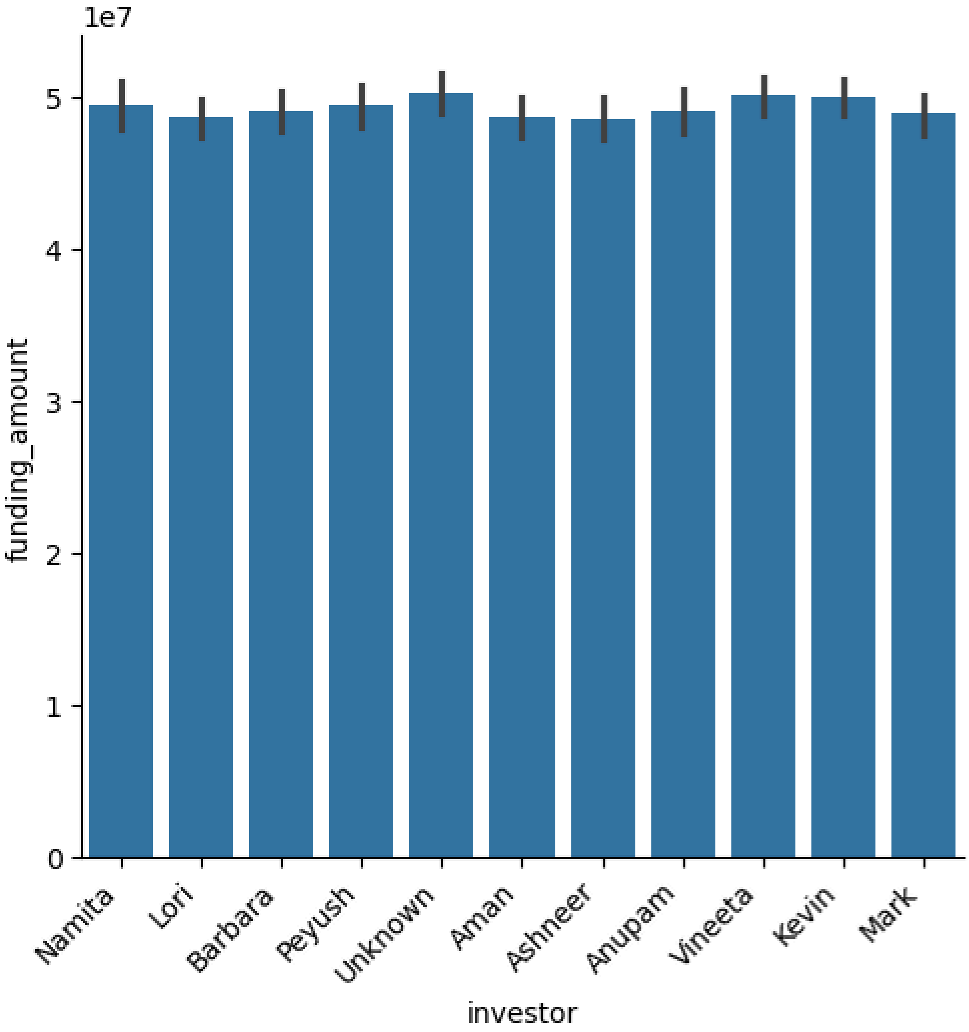
In [238…

```
sns.catplot(x="deal_status",y="funding_amount",data=df,kind="bar")
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for readability
plt.show()
```



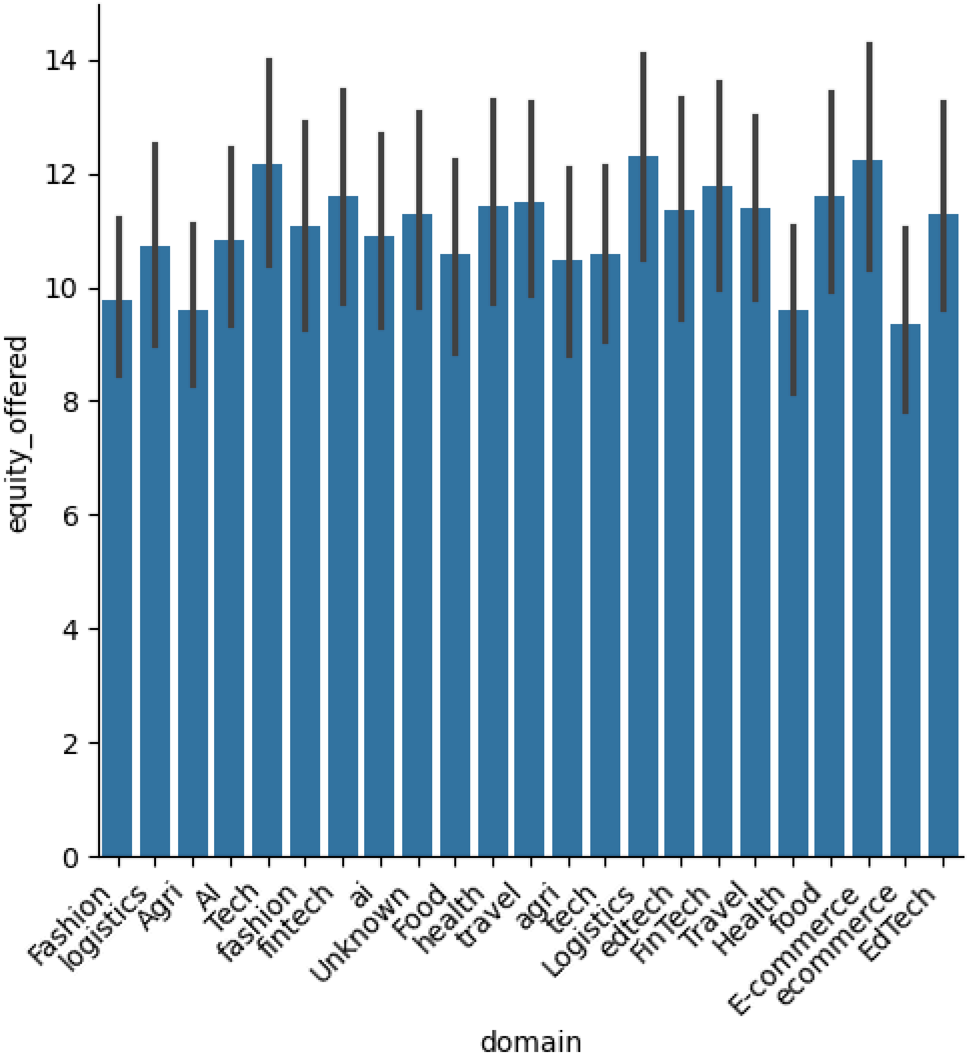
In [239…

```
sns.catplot(x="investor",y="funding_amount",data=df,kind="bar")
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for readability
plt.show()
```



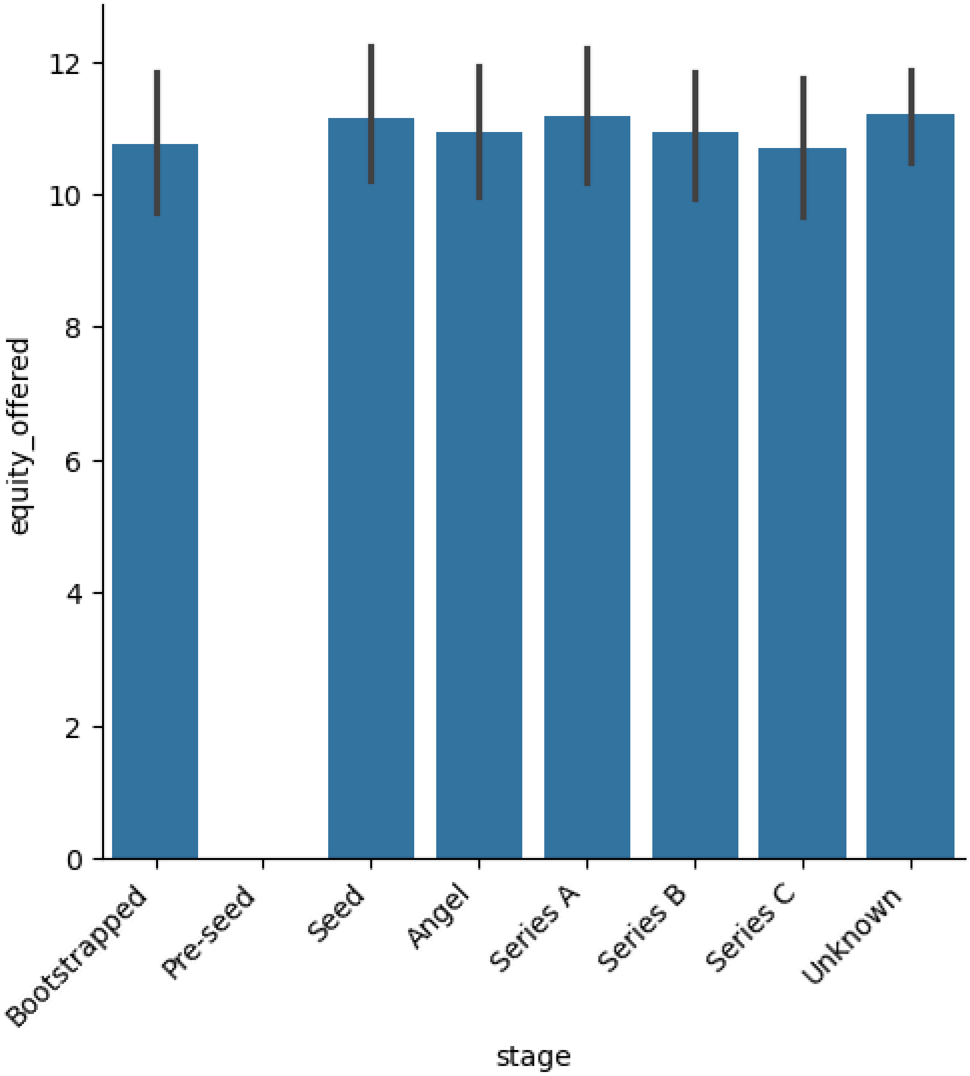
In [240…

```
sns.catplot(x="domain",y="equity_offered",data=df,kind="bar")
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for readability
plt.show()
```

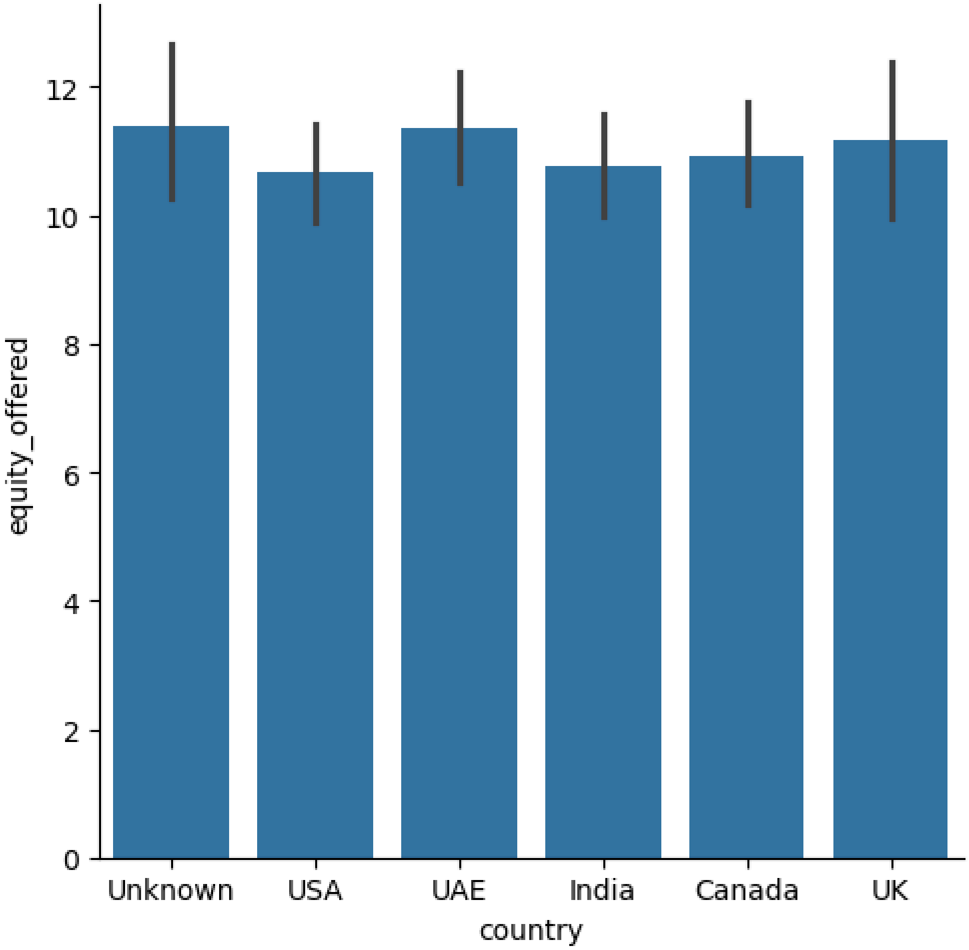


In [241…

```
sns.catplot(x="stage",y="equity_offered",data=df,kind="bar")
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for readability
plt.show()
```

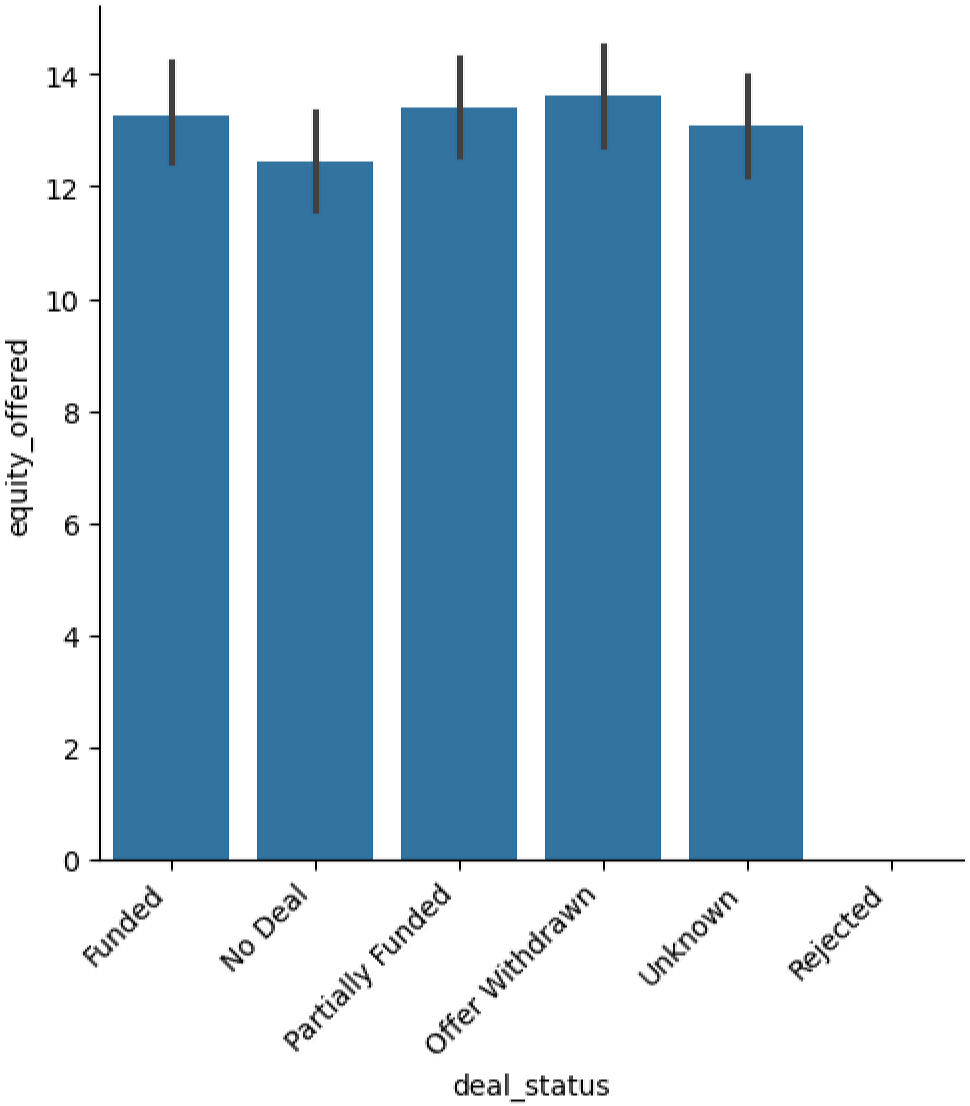


```
In [242…  sns.catplot(x="country",y="equity_offered",data=df,kind="bar")
          plt.show()
```



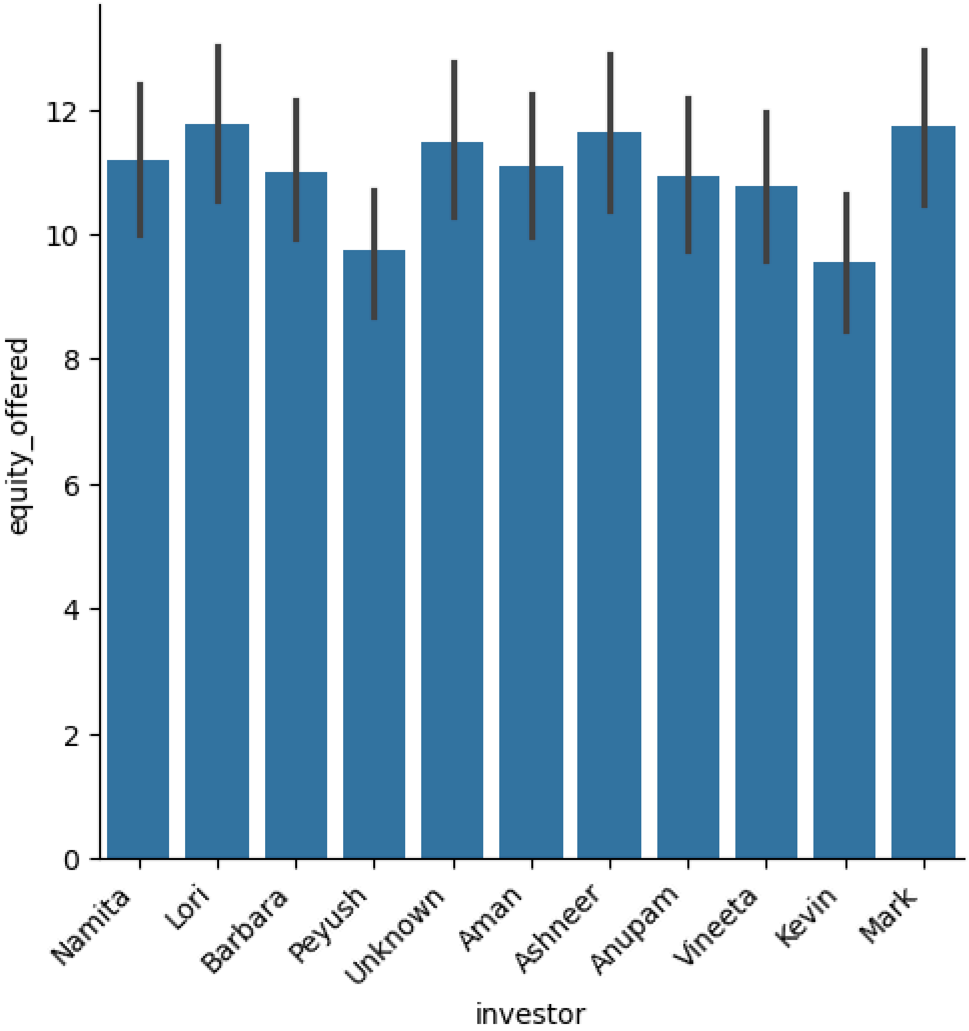
In [243…

```
sns.catplot(x="deal_status",y="equity_offered",data=df,kind="bar")
plt.xticks(rotation=45, ha='right')  # Rotate x-axis labels for readability
plt.show()
```



In [244…

```
sns.catplot(x="investor",y="equity_offered",data=df,kind="bar")
plt.xticks(rotation=45, ha='right')  # Roequity_offeredtate x-axis labels for reada
plt.show()
```



Heat Map

- A heat map in Python is a graphical representation of data where values are depicted using colors

In [246…

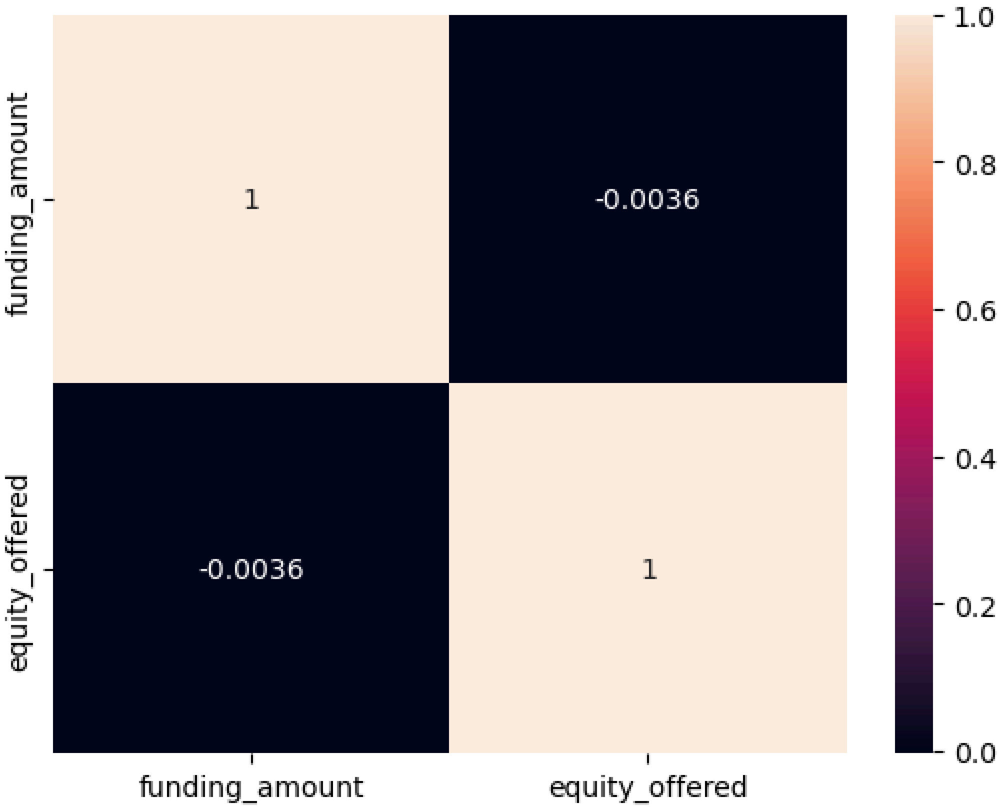
```
hm = df[["funding_amount","equity_offered"]].corr()
hm
```

Out[246…

	funding_amount	equity_offered
funding_amount	1	-0
equity_offered	-0	1

In [247…

```
sns.heatmap(hm,annot=True)
plt.show()
```



Multivariate Measures

- correlation (All Continuous variable)
- crosstab (All discrete variable)
- groupby (2 discrete + 1 continous)

Crosstab

In [250…

```
pd.crosstab([df['investor'], df['stage']], df['domain'],margins = True).head(20)
```

Out[250…

	domain	AI	Agri	EdTech	E-commerce	Fashion	FinTech	Food	Health	L
investor	**stage**									
Aman	**Bootstrapped**	2	2	4	3	3	4	2	3	
	Seed	3	3	1	3	4	1	0	2	
	Angel	1	1	1	0	1	5	2	3	
	Series A	2	2	2	2	2	2	1	1	
	Series B	1	0	1	2	5	1	6	1	
	Series C	3	5	0	0	2	1	4	1	
	Unknown	3	6	7	1	4	8	2	2	
Anupam	**Bootstrapped**	0	3	2	4	5	3	0	1	
	Seed	2	3	1	3	2	0	4	1	
	Angel	1	3	3	4	4	3	2	2	
	Series A	3	4	1	5	2	2	2	4	
	Series B	5	1	1	4	0	1	1	3	
	Series C	2	2	0	1	2	3	2	1	
	Unknown	4	11	6	6	3	5	0	1	
Ashneer	**Bootstrapped**	1	5	3	2	1	5	2	2	
	Seed	2	3	4	1	5	0	3	2	
	Angel	2	3	0	3	3	4	2	5	
	Series A	3	4	1	5	0	1	2	1	
	Series B	0	2	3	2	5	2	5	1	
	Series C	5	1	1	0	2	1	3	3	

20 rows × 24 columns

In [251…

```
pd.crosstab([df['investor'], df['deal_status']], df['country'],margins = True).head
```

Out[251…

	country	Canada	India	UAE	UK	USA	Unknown	All
investor	deal_status							
Aman	Funded	17	18	14	6	14	7	76
	No Deal	12	19	10	7	13	14	75
	Offer Withdrawn	13	18	17	4	14	7	73
	Partially Funded	18	16	11	6	15	9	75
	Rejected	17	9	25	9	11	8	79
	Unknown	15	6	12	9	17	12	71
Anupam	Funded	14	16	19	4	18	3	74
	No Deal	14	13	13	7	11	9	67
	Offer Withdrawn	8	19	17	8	15	9	76
	Partially Funded	14	17	16	3	13	4	67
	Rejected	22	17	17	6	13	5	80
	Unknown	18	16	12	6	13	3	68
Ashneer	Funded	21	19	17	8	17	6	88
	No Deal	11	17	17	6	14	6	71
	Offer Withdrawn	16	20	20	11	14	8	89
	Partially Funded	14	21	14	10	8	5	72
	Rejected	9	14	17	8	17	9	74
	Unknown	19	11	10	5	14	10	69
Barbara	Funded	18	27	15	10	15	3	88
	No Deal	9	19	17	6	20	9	80

GroupBy

In [253…

```
df.groupby(["domain","startup_name"])["funding_amount"].describe().T
```

Out[253...

domain						
startup_name	**AIBox_1855**	**AICore_4377**	**AIFoods_3074**	**AIGen_2246**	**AIGen_998**	**AIHive_24**
count	1	1	1	1	1	
mean	32,617,072	49,114,685	49,114,685	46,043,710	49,114,685	49,114,6
std	NaN	NaN	NaN	NaN	NaN	N
min	32,617,072	49,114,685	49,114,685	46,043,710	49,114,685	49,114,6
25%	32,617,072	49,114,685	49,114,685	46,043,710	49,114,685	49,114,6
50%	32,617,072	49,114,685	49,114,685	46,043,710	49,114,685	49,114,6
75%	32,617,072	49,114,685	49,114,685	46,043,710	49,114,685	49,114,6
max	32,617,072	49,114,685	49,114,685	46,043,710	49,114,685	49,114,6

8 rows × 5000 columns

In [254...

```
df.groupby(["country","stage"])["funding_amount"].describe().T
```

```
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_19244\2143453640.py:1: FutureWarning:
The default of observed=False is deprecated and will be changed to True in a future
version of pandas. Pass observed=False to retain current behavior or observed=True t
o adopt the future default and silence this warning.
  df.groupby(["country","stage"])["funding_amount"].describe().T
```

Out[254...

country	Canada						
stage	**Bootstrapped**	**Seed**	**Angel**	**Series A**	**Series B**	**Series C**	**Unknown**
count	114	126	125	131	129	109	270
mean	50,483,957	48,289,632	48,280,406	50,645,491	49,595,308	50,651,185	48,603,186
std	15,271,681	15,821,424	12,487,087	16,166,228	14,537,067	12,496,254	13,787,950
min	2,724,499	4,569,938	1,663,836	813,173	4,663,470	911,229	498,126
25%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
50%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
75%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
max	99,474,284	98,502,738	99,202,099	99,698,789	96,093,502	99,948,211	98,783,538

8 rows × 42 columns

In [255...

```
df.groupby(["deal_status","investor"])["funding_amount"].describe().T
```

Out[255…

deal_status							
investor	**Aman**	**Anupam**	**Ashneer**	**Barbara**	**Kevin**	**Lori**	**Mark**
count	76	74	88	88	77	72	84
mean	49,379,865	51,608,236	47,605,022	49,701,772	49,888,751	50,229,454	48,093,564
std	17,461,631	11,994,732	16,107,187	16,867,809	13,750,689	11,711,124	11,706,125
min	1,810,637	12,502,745	1,958,965	203,969	4,013,574	177,420	6,075,980
25%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
50%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
75%	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685	49,114,685
max	98,829,135	96,358,130	97,039,592	99,674,401	99,965,336	99,948,211	90,358,929

8 rows × 66 columns

In [256…

```
df.groupby(["domain","startup_name"])["equity_offered"].describe().T
```

Out[256…

domain						
startup_name	**AIBox_1855**	**AICore_4377**	**AIFoods_3074**	**AIGen_2246**	**AIGen_998**	**AIHive_24**
count	1	1	1	1	1	
mean	7	7	7	7	13	
std	NaN	NaN	NaN	NaN	NaN	N
min	7	7	7	7	13	
25%	7	7	7	7	13	
50%	7	7	7	7	13	
75%	7	7	7	7	13	
max	7	7	7	7	13	

8 rows × 5000 columns

In [257…

```
df.groupby(["country","stage"])["equity_offered"].describe().T
```

```
C:\Users\WELCOME\AppData\Local\Temp\ipykernel_19244\2734522.py:1: FutureWarning: The
default of observed=False is deprecated and will be changed to True in a future vers
ion of pandas. Pass observed=False to retain current behavior or observed=True to ad
opt the future default and silence this warning.
  df.groupby(["country","stage"])["equity_offered"].describe().T
```

Out[257…

country							Canada		
stage	Bootstrapped	Seed	Angel	Series A	Series B	Series C	Unknown	Bootstrapped	Seed
count	114	126	125	131	129	109	270	138	137
mean	9	10	11	11	10	12	12	10	11
std	10	11	13	12	12	14	14	11	13
min	0	0	0	0	0	0	0	0	0
25%	7	7	7	7	7	7	7	7	7
50%	7	7	7	7	7	7	7	7	7
75%	7	7	7	7	7	7	7	7	7
max	51	53	59	58	53	59	59	57	56

8 rows × 42 columns

In [265…

```
df.groupby(["deal_status","investor"])["equity_offered"].describe().T
```

Out[265…

deal_status									Fur
investor	Aman	Anupam	Ashneer	Barbara	Kevin	Lori	Mark	Namita	Peyush	Unkn
count	76	74	88	88	77	72	84	79	74	
mean	14	12	14	14	11	12	14	16	10	
std	13	12	14	14	9	12	14	16	9	
min	3	1	1	2	1	3	1	7	4	
25%	7	7	7	7	7	7	7	7	7	
50%	7	7	7	7	7	7	7	7	7	
75%	17	7	10	7	7	7	17	20	7	
max	59	57	59	59	53	59	59	57	56	

8 rows × 66 columns