Assignment 1: Wrangling and EDA

Foundations of Machine Learning

Q1. This question provides some practice cleaning variables which have common problems.

1. Numeric variable: For `airbnb_NYC.csv`, clean the `Price` variable as well as you can, and explain the choices you make. How many missing values do you end up with? (Hint: What happens to the formatting when a price goes over 999 dollars, say from 675 to 1,112?)
2. Categorical variable: For the Minnesota police use of for data, `mn_police_use_of_force.csv`, clean the `subject_injury` variable, handling the NA's; this gives a value `Yes` when a person was injured by police, and `No` when no injury occurred. What proportion of the values are missing? Cross-tabulate your cleaned `subject_injury` variable with the `force_type` variable. Are there any patterns regarding when the data are missing? For the remaining missing values, replace the `np.nan/None` values with the label `Missing`.
3. Dummy variable: For `metabric.csv`, convert the `Overall Survival Status` variable into a dummy/binary variable, taking the value 0 if the patient is deceased and 1 if they are living.
4. Missing values: For `airbnb_NYC.csv`, determine how many missing values of `Review Scores Rating` there are. Create a new variable, in which you impute the median score for non-missing observations to the missing ones. Why might this bias or otherwise negatively impact your results?

Question 1

In [17]:
```
import pandas as pd

df = pd.read_csv('airbnb_NYC.csv', encoding='latin-1')
df.head()
```

Out[17]:

	Host Id	Host Since	Name	Neighbourhood	Property Type	Review Scores Rating (bin)	Room Type	Zipco
0	5162530	NaN	1 Bedroom in Prime Williamsburg	Brooklyn	Apartment	NaN	Entire home/apt	1124
1	33134899	NaN	Sunny, Private room in Bushwick	Brooklyn	Apartment	NaN	Private room	1120
2	39608626	NaN	Sunny Room in Harlem	Manhattan	Apartment	NaN	Private room	1003
3	500	6/26/2008	Gorgeous 1 BR with Private Balcony	Manhattan	Apartment	NaN	Entire home/apt	1002
4	500	6/26/2008	Trendy Times Square Loft	Manhattan	Apartment	95.0	Private room	1003

In [18]:
```
df['Price'].value_counts().head(30)
```

Out[18]:
```
Price
150    1481
100    1207
200    1059
125     889
75      873
80      798
250     747
120     743
90      729
70      711
175     705
65      696
60      683
50      643
85      623
95      558
99      558
110     541
140     457
130     457
160     449
55      437
180     399
300     397
225     384
135     373
199     353
115     334
45      324
195     298
Name: count, dtype: int64
```

In [19]:
```
df['Price'].unique()[:50]
```

Out[19]:
```
array(['145', '37', '28', '199', '549', '149', '250', '90', '270', '290',
       '170', '59', '49', '68', '285', '75', '100', '150', '700', '125',
       '175', '40', '89', '95', '99', '499', '120', '79', '110', '180',
       '143', '230', '350', '135', '85', '60', '70', '55', '44', '200',
       '165', '115', '74', '84', '129', '50', '185', '80', '190', '140'],
      dtype=object)
```

In [21]:
```
df['Price'] = df['Price'].astype(str)
df['Price'] = df['Price'].str.replace('$', '').str.replace(',', '')
df['Price'] = pd.to_numeric(df['Price'], errors='coerce')
```

In [23]:
```
df['Price'].isna().sum()
```

Out[23]:
```
0
```

Answer to Question 1

I chose to remove the dollar sign and commas, from the "Price" column because they prevent the computer from doing actual calculations. If I did not remove the comma, any price over 999 (like for example 1112) would have automatically been considered a "missing value" because the computer would not identify it as a number. By cleaning these symbols first and then using "to_numeric", I was able to convert the entire column effectively (from just text to floats). And finally after running my code, I ended up with zero missing values --> every number (price) in the dataset is now ready for mathematical analysis.

Question 2

In [24]:
```
df = pd.read_csv('mn_police_use_of_force.csv')
```

In [25]:
```
print("Proportion of missing values:")
print(df['subject_injury'].isna().mean())
```

```
Proportion of missing values:
0.7619342359767892
```

In [27]:
```
print("Comparison Table (Force Type vs Injury):")
print(pd.crosstab(df['force_type'], df['subject_injury'], dropna=False))
```

```
Comparison Table (Force Type vs Injury):
subject_injury            No   Yes
force_type
Baton                      0     2
Bodily Force            1093  1286
Chemical Irritant        131    41
Firearm                    2     0
Gun Point Display         33    44
Improvised Weapon         34    40
Less Lethal Projectile     1     2
Police K9 Bite             2    44
Taser                    150   172
```

In [28]:
```
df['subject_injury'] = df['subject_injury'].fillna('Missing')
```

In [31]:
```
print(f'New counts with "Missing" label:')
print(df['subject_injury'].value_counts())
```

```
New counts with "Missing" label:
subject_injury
Missing    9848
Yes        1631
No         1446
Name: count, dtype: int64
```

Answer to Question 2

About 76% of the 'subject_injury' data is missing. When looking at the comparison table, a pattern definitely emerges --> the injury status is often left blank in many of the force categories. To fix this, I replaced all the empty values with the label "Missing". This will now allow me to keep all the records in my analysis while clearly showing which incidents did not have an injury report filed.

Question 3

In [37]:
```
df_3['Survival_Binary'] = 0
df_3.loc[df_3['Overall Survival Status'] == '0:LIVING', 'Survival_Binary'] = 1
print(df_3[['Overall Survival Status', 'Survival_Binary']].head())
```

```
  Overall Survival Status  Survival_Binary
0               0:LIVING                1
1             1:DECEASED                0
2               0:LIVING                1
3             1:DECEASED                0
4             1:DECEASED                0
```

Question 4

In [40]:
```
import pandas as pd
df_1 = pd.read_csv('airbnb_NYC.csv', encoding='latin1')
print("Missing values in Review Scores Rating:")
print(df_1['Review Scores Rating'].isna().sum())
```

```
Missing values in Review Scores Rating:
8323
```

In [42]:
```
median_score = df_1['Review Scores Rating'].median()
df_1['Review_Scores_Imputed'] = df_1['Review Scores Rating'].fillna(median_score)

print("Missing values after filling:")
print(df_1['Review_Scores_Imputed'].isna().sum())
```

```
Missing values after filling:
0
```

Q2. Go to https://sharkattackfile.net/ and download their dataset on shark attacks.

1. Open the shark attack file using Pandas. It is probably not a csv file, so `read_csv` won't work. What does work?
2. Drop any columns that do not contain data.
3. What is an observation? Carefully justify your answer, and explain how it affects your choices in cleaning and analyzing the data.
4. Clean the year variable. Describe the range of values you see. Filter the rows to focus on attacks since 1940. Are attacks increasing, decreasing, or remaining constant over time?
5. Clean the Age variable and make a histogram of the ages of the victims.

6. Clean the `Type` variable so it only takes three values: Provoked and Unprovoked and Unknown. What proportion of attacks are unprovoked?
7. Clean the `Fatal Y/N` variable so it only takes three values: Y, N, and Unknown.
8. Is the attack more or less likely to be fatal when the attack is provoked or unprovoked? Thoughts?

Question 1

In [50]:
```
!pip install xlrd
```

```
Defaulting to user installation because normal site-packages is not writeable
Looking in links: /usr/share/pip-wheels
Requirement already satisfied: xlrd in /home/80053afd-fb7a-4c88-9252-538fd83b7785/.l
ocal/lib/python3.11/site-packages (2.0.2)
```

In [1]:
```
import pandas as pd
df_shark = pd.read_excel('GSAF5.xls')
df_shark.head()
```

Out[1]:

	Date	Year	Type	Country	State	Location	Activity	Name	Sex
0	29th January	2026.0	Unprovoked	Brazil	Recife	Del Chifre Beach in Olinda	Swimming	Deivson Rocha Dantas	M
1	29th January	2026.0	Unprovoked	Australia	NSW	Angels Beach East Ballina	Surfing	Unnamed man	M
2	24th January	2026.0	Unprovoked	Australia	Tasmania	Cooee Beach west of Burnie	Swimming	Megan Stokes	F
3	20th January	2026.0	Unprovoked	Australia	NSW	Point Plomber North of Port Macquarie	Surfing	Paul Zvirdinas	M
4	19th January	2026.0	Unprovoked	Australia	NSW	Dee Why	Surfing	Unknown	M

5 rows × 23 columns

Answer for Question 1

Since the shark attack data is saved as an '.xls' and not a plain text file, the usual read_csv() will not work. Instead, we can use pd.read_excel() to correctly use the file in our notebook.

Question 2

In [3]:
```
df_shark = df_shark.dropna(axis=1, how='all')
print(f"Columns that remain after dropping any columns that do not contain data: {l
```

```
Columns that remain after dropping any columns that do not contain data: 23
```

Question 3

In this dataset, an observation is a single row representing one specific shark attack incident. Since each observation records one unique event, it is very important to check for duplicate rows to make sure the same incident is not counted twice. This will also help when we clean our data, as we have to decide whether an observation with missing details should be kept or removed. So, by treating each row as its own unique event, I can do multiple kinds of analysis on the dataset.

Question 4

In [5]:
```
df_shark['Year'] = df_shark['Year'].fillna(0)
print("Year Range Statistics:")
print(df_shark['Year'].describe())
```

```
Year Range Statistics:
count    7073.000000
mean     1935.444083
std       272.601371
min         0.000000
25%      1948.000000
50%      1986.000000
75%      2010.000000
max      2026.000000
Name: Year, dtype: float64
```

The table shows that shark attacks are becoming more frequent because the years listed for each quarter of the data are getting closer together. It took almost all of history to reach the first 25% of attacks by 1948, but it only took 24 years to move from the 50% mark (in 1986) to the 75% mark (in 2010). These decreasing gaps prove that a huge portion of the shark attacks are happening in recent years.

In [6]:
```
df_since_1940 = df_shark[df_shark['Year'] >= 1940]
print("First 5 rows of the filtered data:")
print(df_since_1940[['Date', 'Year', 'Country']].head())
```

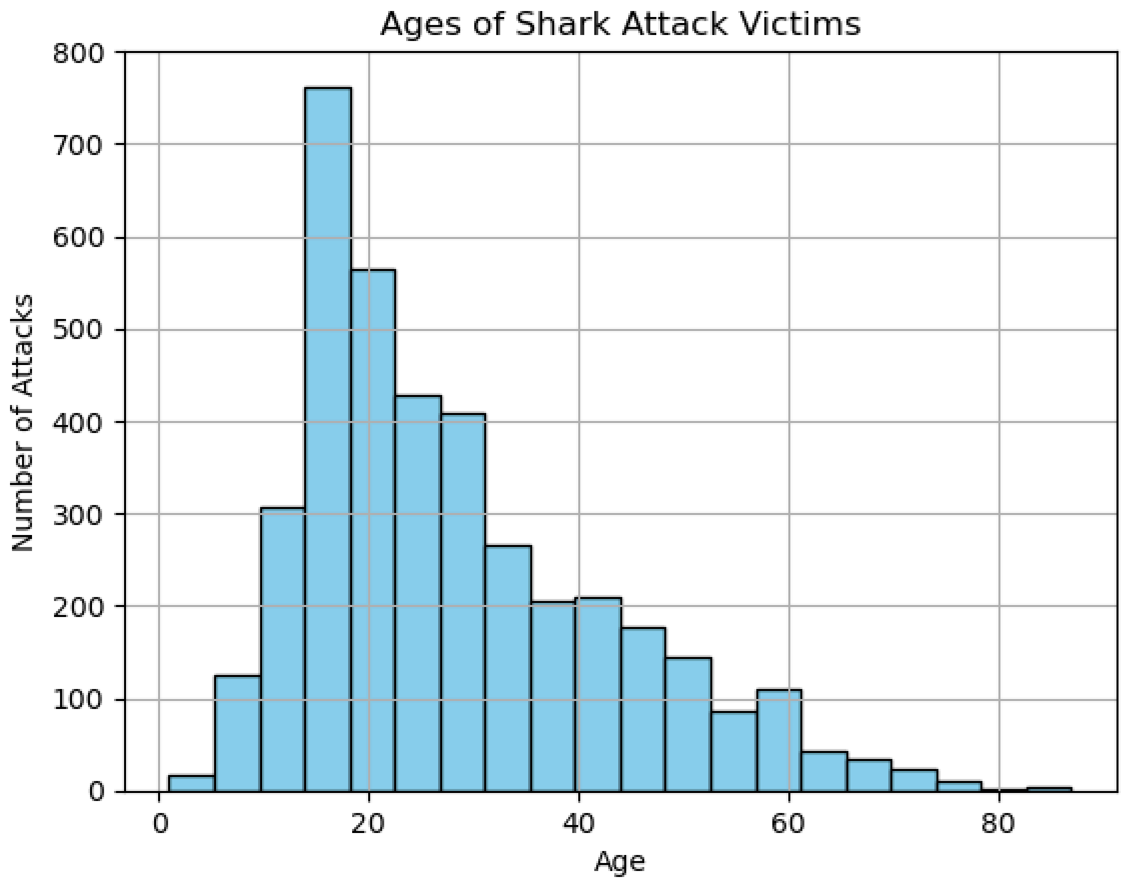
```
First 5 rows of the filtered data:
          Date    Year    Country
0  29th January  2026.0     Brazil
1  29th January  2026.0  Australia
2  24th January  2026.0  Australia
3  20th January  2026.0  Australia
4  19th January  2026.0  Australia
```

Answer to Question 4

I cleaned the 'year' variable to handle any missing values so they would not interfere with analysis. The data covers a very wide range of time, so by applying a filter to select only rows where the year is 1940 or later, I can focus the dataset on more modern incidents. After looking at these filtered results, I can observe that shark attacks appear to be increasing over time, most likely due to an increase in beach tourism.

Question 5

In [9]:
```
df_shark['Age'] = pd.to_numeric(df_shark['Age'], 'coerce')

import matplotlib.pyplot as plt
df_shark['Age'].dropna().hist(bins=20, color='skyblue', edgecolor='black')
plt.title('Ages of Shark Attack Victims')
plt.xlabel('Age')
plt.ylabel('Number of Attacks')
plt.show()
```



Question 6

In [10]:
```
valid_types = ['Provoked', 'Unprovoked']
df_shark.loc[df_shark['Type'].isin(valid_types) == False, 'Type'] = 'Unknown'
print(df_shark['Type'].value_counts(normalize=True))
```

```
Type
Unprovoked    0.738583
Unknown       0.170649
Provoked      0.090768
Name: proportion, dtype: float64
```

Answer to Question 6

After cleaning the data, I made sure "Provoked" and "Unprovoked" remained, while labeling all other categories as "Unknown". My analysis tells me that unprovoked attacks represent the largest population of recorded incidents.

Question 7

In [14]:
```
print(df_shark.columns)
```

```
Index(['Date', 'Year', 'Type', 'Country', 'State', 'Location', 'Activity',
       'Name', 'Sex', 'Age', 'Injury', 'Fatal Y/N', 'Time', 'Species ',
       'Source', 'pdf', 'href formula', 'href', 'Case Number', 'Case Number.1',
       'original order', 'Unnamed: 21', 'Unnamed: 22'],
      dtype='object')
```

In [16]:
```
valid_fatal = ['Y', 'N']

df_shark.loc[df_shark['Fatal Y/N'].isin(valid_fatal) == False, 'Fatal Y/N'] = 'Unkn
print(df_shark['Fatal Y/N'].value_counts())
```

```
Fatal Y/N
N          4932
Y          1488
Unknown     653
Name: count, dtype: int64
```

In [18]:
```
pd.crosstab(df_shark['Type'], df_shark['Fatal Y/N'])
```

Out[18]:

Fatal Y/N Type	N	Unknown	Y
Provoked	610	12	20
Unknown	451	555	201
Unprovoked	3871	86	1267

Question 8

According to my investigation, attacks are less likely to be fatal when they are provoked, as these incidents usually involve defensive bites during activities such as fishing or handling. Unprovoked attacks have a higher fatality rate because this is when sharks will demonstrate predatory behaviors on swimmers or surfers.

Q3. Open the "tidy_data.pdf" document available in `https://github.com/ds4e/wrangling`, which is a paper called *Tidy Data* by Hadley Wickham.

1. Read the abstract. What is this paper about?
2. Read the introduction. What is the "tidy data standard" intended to accomplish?
3. Read the intro to section 2. What does this sentence mean: "Like families, tidy datasets are all alike but every messy dataset is messy in its own way." What does this sentence mean: "For a given dataset, it's usually easy to figure out what are observations and what are variables, but it is surprisingly difficult to precisely define variables and observations in general."
4. Read Section 2.2. How does Wickham define values, variables, and observations?

5. How is "Tidy Data" defined in section 2.3?
6. Read the intro to Section 3 and Section 3.1. What are the 5 most common problems with messy datasets? Why are the data in Table 4 messy? What is "melting" a dataset?
7. Why, specifically, is table 11 messy but table 12 tidy and "molten"?

Question 1

This paper explains a better way to organize information so that computers and people can understand it easily. Usually, a lot of time is wasted fixing messy date before any real work can begin. The author calls his solution "Tidy Data", which is a specific way to structure tables. In a tidy dataset, every column is a variable and every row is a single observation. This setup makes it much faster to create charts, run mathematical calculations, and look for patterns. Instead of learning new tricks for every new project, you can use the same small set of tools every time. By following these rules, data scientists can spend less time cleaning and more time finding solutions.

Question 2

The "tidy data standard" is a set of rules used to organize information so that it is easy to explore and analyze. It is designed to stop researchers from having to "reinvent the wheel" every time they start a new project with messy files. By using a "standard structure", different computer tools can work together perfectly without needing extra cleaning steps in between.

Question 3

The first sentence means that all clean data follows the same rules, while messy data can be broken and very confusing in thousands of different ways. The second sentence explains that while you can usually spot the patterns in one specific file, it is hard to create a single definitions for all rows and columns that works for every situation.

Question 4

Values are the individual pieces of information in a dataset, like a specific number or a single word. Variables are groups

that contain all the values measuring the same attribute. Observations are groups that contain all the different measurements taken for a single unit, like all the data for one person or one day.

Question 5

"Tidy Data" is defined by three simple rules that link the meaning of data to its physical structure. First, every variable you measure must have its own dedicated column. Second, every individual observation or "case" must have its own dedicated row. Third, each different type of experimental unit should be stored in its own separate table.

Question 6

1. Column headers are values, not variable names

2. Multiple variables are stored in one column

3. Variables are stored in both rows and columns

4. Multiple types of observational units are stored in the same table

5. A single observational unit is stored in multiple tables

Table 4 is messy because the column headers represent specific income values, rather than a single variable name.

"Melting" is a process that turns columns into rows to organize the data better, transforming a wide table into "tidy one".

Question 7

Table 11 is messy because it spreads the "day" variable across multiple columns and stores actual variable names like temperature in a column called "element".

Table 12 is tidy and "molten" because it uses a standard layout where each column is a single variable and each row is a single day's observation.

Q4. This question looks at financial transfers from international actors to American universities. In particular, from which countries and giftors are the gifts coming from, and to which institutions are they going?

For this question, `.groupby([vars]).count()` and `.groupby([vars]).sum()` will be especially useful to tally the number of occurrences and sum the values of those occurrences.

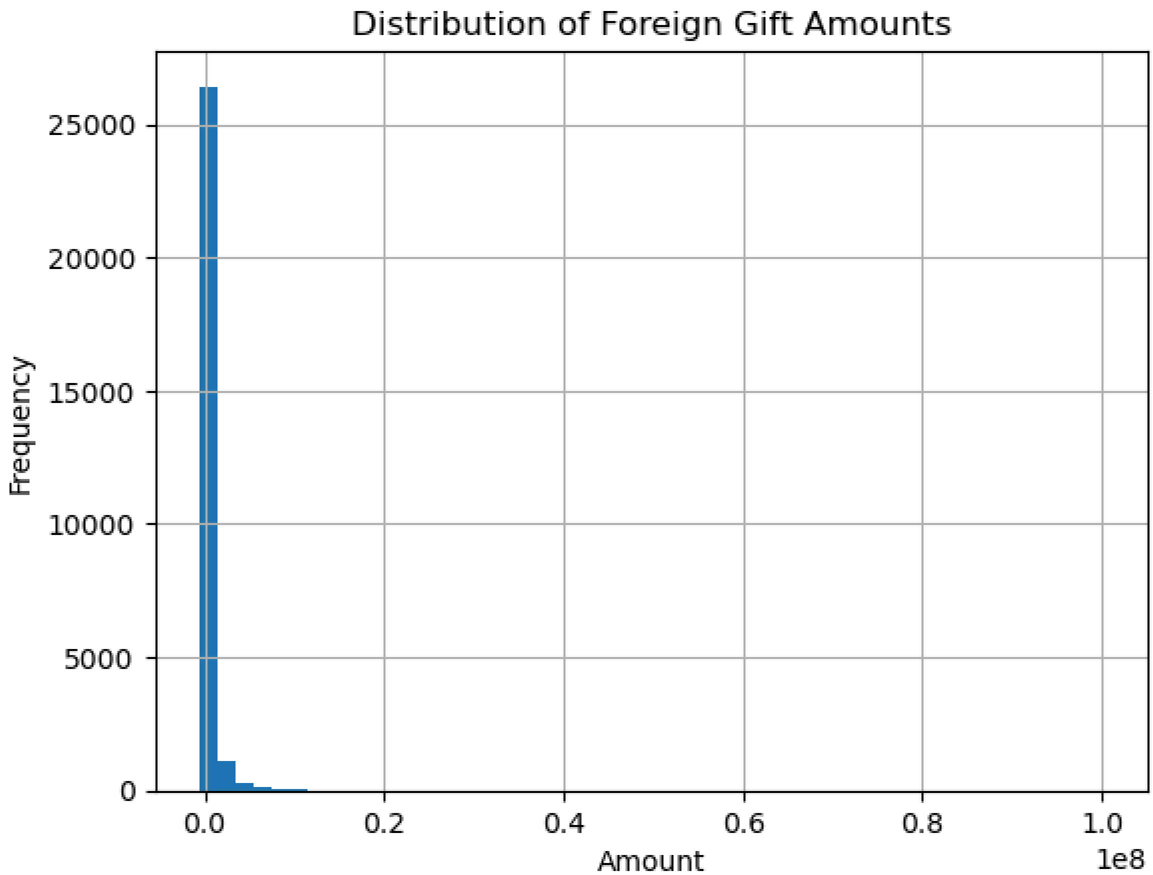
1. Load the `ForeignGifts_edu.csv` dataset.
2. For `Foreign Gift Amount`, create a histogram and describe the variable. Describe your findings.
3. For `Gift Type`, create a histogram or value counts table. What proportion of the gifts are contracts, real estate, and monetary gifts?
4. What are the top 15 countries in terms of the number of gifts? What are the top 15 countries in terms of the amount given?
5. What are the top 15 institutions in terms of the total amount of money they receive? Make a histogram of the total amount received by all institutions.
6. Which giftors provide the most money, in total?

Question 1

In [3]:
```
import pandas as pd
df_gifts = pd.read_csv('ForeignGifts_edu.csv')
```

Question 2

In [7]:
```
import matplotlib.pyplot as plt
df_gifts['Foreign Gift Amount'].hist(bins=50)
plt.title('Distribution of Foreign Gift Amounts')
plt.xlabel('Amount')
plt.ylabel('Frequency')
plt.show()
```



Answer for Question 2

The "Foreign Gift Amount" variable is highly right-skewed, meaning there are many small gifts and only a few extremely larges ones that stretch the scale. Most of the data is tightly packed between 0.0 and 0.2, which shows that small-scale donations happen much more than multi million dollar ones.

Question 3

In [8]:
```
gift_proportions = df_gifts['Gift Type'].value_counts(normalize=True)
print(gift_proportions)
```

```
Gift Type
Contract         0.612097
Monetary Gift    0.387513
Real Estate      0.000390
Name: proportion, dtype: float64
```

Answer for Question 3

Contracts make up the largest share about 61%, while Monetary Gifts account for nearly 39%. Real estate is

extremely rare, representing less than 0.1% of all gifts in the dataset.

Question 4

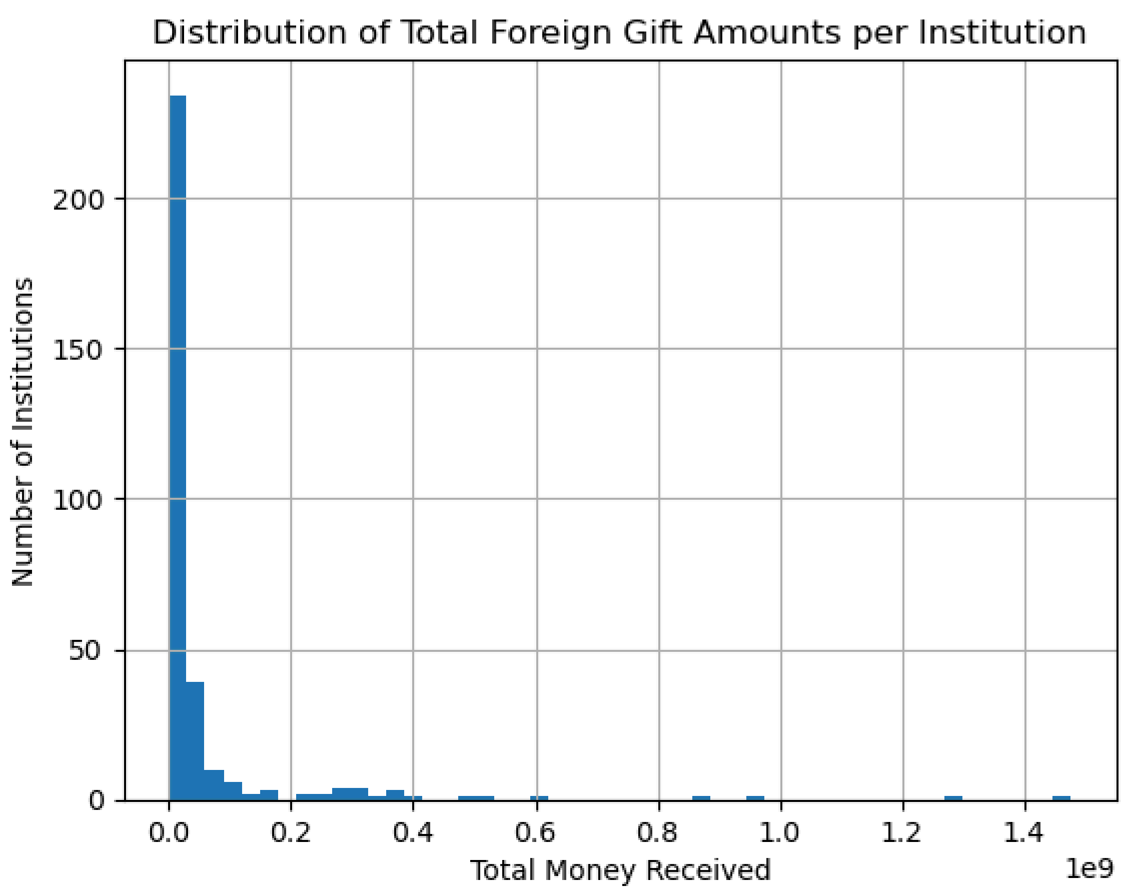
In [12]:
```
top_15_count = df_gifts['Country of Giftor'].value_counts().head(15)
print("Top 15 by Number of Gifts:")
print(top_15_count)

top_15_amount = df_gifts.groupby('Country of Giftor')['Foreign Gift Amount'].sum().
print("\nTop 15 by Total Amount:")
print(top_15_amount)
```

```
Top 15 by Number of Gifts:
Country of Giftor
ENGLAND            3655
CHINA              2461
CANADA             2344
JAPAN              1896
SWITZERLAND        1676
SAUDI ARABIA       1610
FRANCE             1437
GERMANY            1394
HONG KONG          1080
SOUTH KOREA         811
QATAR               693
THE NETHERLANDS     512
KOREA               452
INDIA               434
TAIWAN              381
Name: count, dtype: int64

Top 15 by Total Amount:
Country of Giftor
QATAR                   2706240869
ENGLAND                 1464906771
CHINA                   1237952112
SAUDI ARABIA            1065205930
BERMUDA                  899593972
CANADA                   898160656
HONG KONG                887402529
JAPAN                    655954776
SWITZERLAND              619899445
INDIA                    539556490
GERMANY                  442475605
UNITED ARAB EMIRATES     431396357
FRANCE                   405839396
SINGAPORE                401157692
AUSTRALIA                248409202
Name: Foreign Gift Amount, dtype: int64
```

Question 5

In [13]:
```
inst_totals = df_gifts.groupby('Institution Name')['Foreign Gift Amount'].sum().sor

print("Top 15 Institutions by Total Amount:")
print(inst_totals.head(15))

inst_totals.hist(bins=50)
plt.title('Distribution of Total Foreign Gift Amounts per Institution')
plt.xlabel('Total Money Received')
plt.ylabel('Number of Institutions')
plt.show()
```

```
Top 15 Institutions by Total Amount:
Institution Name
Carnegie Mellon University                       1477922504
Cornell University                               1289937761
Harvard University                                954803610
Massachusetts Institute of Technology             859071692
Yale University                                   613441311
Texas A&M University                              521455050
Johns Hopkins University                          502409595
Northwestern University                           402316221
Georgetown University                             379950511
University of Chicago (The)                       364544338
University of Colorado Boulder                    360173159
Duke University                                   343699498
Brigham Young University                          323509863
Stanford University                               319561362
University of Texas MD Anderson Cancer Center     301527419
Name: Foreign Gift Amount, dtype: int64
```



Question 6

In [15]:
```
top_giftors = df_gifts.groupby('Giftor Name')['Foreign Gift Amount'].sum().sort_val
print(top_giftors.head(5))
```

```
Giftor Name
Qatar Foundation                          1166503744
Qatar Foundation/Qatar National Res        796197000
Qatar Foundation for Education             373945215
Anonymous                                  338793629
Saudi Arabian Cultural Mission             275221475
Name: Foreign Gift Amount, dtype: int64
```

Answer to Question 6

The top donors, such as the Qatar Foundation, provide the largest total contributions to American universities.

Q5. This question provides some practice doing exploratory data analysis and visualization.

We'll use the `college_completion.csv` dataset from the US Department of Education. The "relevant" variables for this question are:

- `level` - Level of institution (4-year, 2-year)
- `aid_value` - The average amount of student aid going to undergraduate recipients
- `control` - Public, Private not-for-profit, Private for-profit
- `grad_100_value` - percentage of first-time, full-time, degree-seeking undergraduates who complete a degree or certificate program within 100 percent of expected time (bachelor's-seeking group at 4-year institutions)

1. Load the `college_completion.csv` data with Pandas.
2. How many observations and variables are in the data? Use `.head()` to examine the first few rows of data.
3. Cross tabulate `control` and `level`. Describe the patterns you see in words.
4. For `grad_100_value`, create a kernel density plot and describe table. Now condition on `control`, and produce a kernel density plot and describe tables for each type of institutional control. Which type of institution appear to have the most favorable graduation rates?
5. Make a scatterplot of `grad_100_value` by `aid_value`, and compute the covariance and correlation between the two variables. Describe what you see. Now make the same plot and statistics, but conditioning on `control`. Describe what you see. For which kinds of institutions does aid seem to vary positively with graduation rates?

Question 1

In [3]:
```
import pandas as pd
df_college = pd.read_csv('college_completion.csv')
```

Question 2

In [9]:
```
print(df_college.head())
print(df_college.shape)
```

```
   index  unitid                            chronname        city    state  \
0      0  100654               Alabama A&M University      Normal  Alabama   
1      1  100663  University of Alabama at Birmingham  Birmingham  Alabama   
2      2  100690                  Amridge University  Montgomery  Alabama   
3      3  100706  University of Alabama at Huntsville  Huntsville  Alabama   
4      4  100724             Alabama State University  Montgomery  Alabama   

    level                control  \
0  4-year                 Public   
1  4-year                 Public   
2  4-year  Private not-for-profit   
3  4-year                 Public   
4  4-year                 Public   

                                               basic hbcu flagship  ...  \
0  Masters Colleges and Universities--larger prog...    X      NaN  ...   
1  Research Universities--very high research acti...  NaN      NaN  ...   
2            Baccalaureate Colleges--Arts & Sciences  NaN      NaN  ...   
3  Research Universities--very high research acti...  NaN      NaN  ...   
4  Masters Colleges and Universities--larger prog...    X      NaN  ...   

   vsa_grad_after6_transfer  vsa_grad_elsewhere_after6_transfer  \
0                      36.4                                 5.6   
1                       NaN                                 NaN   
2                       NaN                                 NaN   
3                       0.0                                 0.0   
4                       NaN                                 NaN   

  vsa_enroll_after6_transfer  vsa_enroll_elsewhere_after6_transfer  \
0                       17.2                                  11.1   
1                        NaN                                   NaN   
2                        NaN                                   NaN   
3                        0.0                                   0.0   
4                        NaN                                   NaN   

                                             similar  state_sector_ct  \
0  232937|100724|405997|113607|139533|144005|2285...               13   
1  196060|180461|201885|145600|209542|236939|1268...               13   
2  217925|441511|205124|247825|197647|221856|1353...               16   
3  232186|133881|196103|196413|207388|171128|1900...               13   
4  100654|232937|242617|243197|144005|241739|2354...               13   

   carnegie_ct  counted_pct  nicknames  cohort_size  
0          386      99.7|07        NaN        882.0  
1          106      56.0|07        UAB       1376.0  
2          252     100.0|07        NaN          3.0  
3          106      43.1|07        UAH        759.0  
4          386      88.0|07        ASU       1351.0  

[5 rows x 63 columns]
(3798, 63)
```

Question 3

In [10]:
```
cross_table = pd.crosstab(df_college['control'], df_college['level'])
print(cross_table)
```

```
level                    2-year  4-year
control
Private for-profit          465     527
Private not-for-profit       68    1180
Public                      926     632
```

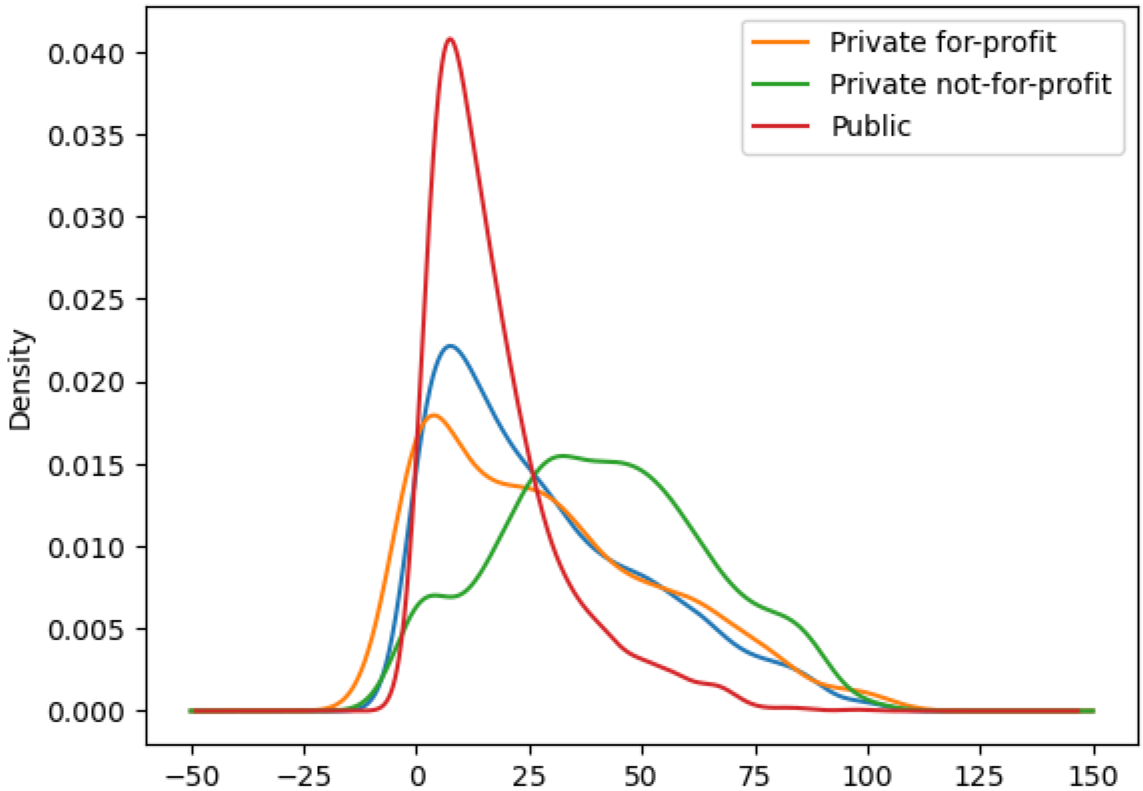
Answer to Question 3

For public schools: Most public institutions are 2-year schools (926) rather 4-year schools (632). For Private Not-for-Profit: This group is overwhelmingly made up of 4-year institutions with a few 2-year options. For Private For-Profit: These schools are more evenly split between 2-year (465) and 4-year (527) levels.

Question 4

In [12]:
```
df_college['grad_100_value'].plot(kind='density')
print(df_college['grad_100_value'].describe())

df_college.groupby('control')['grad_100_value'].plot(kind='density', legend=True)
print(df_college.groupby('control')['grad_100_value'].describe())
```

```
count    3467.000000
mean       28.364465
std        23.312730
min         0.000000
25%         9.000000
50%        22.500000
75%        43.650000
max       100.000000
Name: grad_100_value, dtype: float64
                         count       mean        std  min    25%   50%    75%  \
control
Private for-profit       779.0  29.108858  25.601687  0.0   6.95  24.7  46.75
Private not-for-profit  1189.0  41.660976  23.551231  0.0  25.00  41.0  58.30
Public                  1499.0  17.430887  14.729443  0.0   6.90  13.2  23.25

                          max
control
Private for-profit      100.0
Private not-for-profit  100.0
Public                   97.8
```



Answer in Question 4

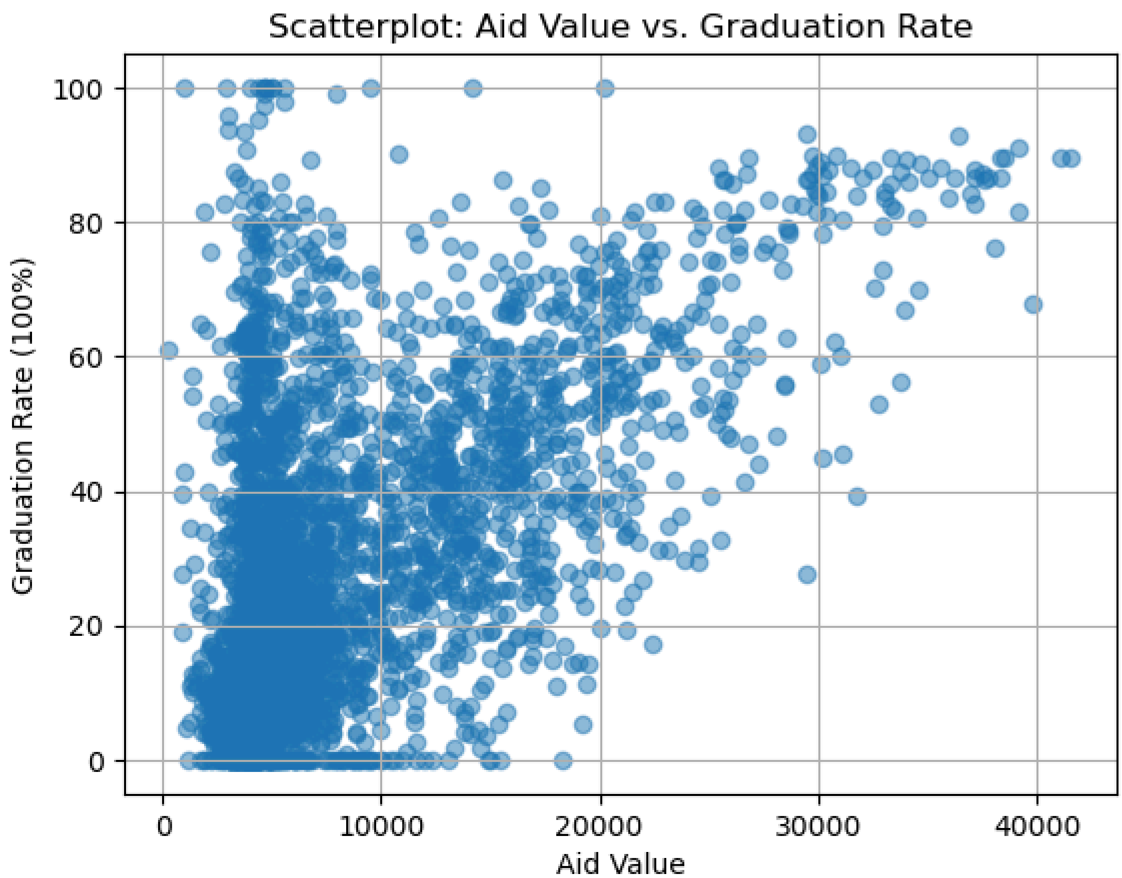
Based on the data for grad_100_value, Private Not-for-Profit appear to have the most favorable graduation rates. They have the highest mean graduation rate compared to Private for-profit & Public institutions. And in the density plot, the green line is shifted further to the right and has a much "fatter" tail toward higher values compared to the other groups, illustrating a higher concentration of students graduating on time.

Question 5

In [15]:
```
import matplotlib.pyplot as plt
plt.scatter(df_college['aid_value'], df_college['grad_100_value'], alpha=0.5)
plt.title('Scatterplot: Aid Value vs. Graduation Rate')
plt.xlabel('Aid Value')
plt.ylabel('Graduation Rate (100%)')
plt.grid(True)

covariance = df_college[['aid_value', 'grad_100_value']].cov()
correlation = df_college[['aid_value', 'grad_100_value']].corr()

print("Covariance")
print(covariance)
```

```
print("\nCorrelation")
print(correlation)
```

```
Covariance
                  aid_value  grad_100_value
aid_value       4.121201e+07    88610.483169
grad_100_value  8.861048e+04      543.483382

Correlation
                aid_value  grad_100_value
aid_value        1.000000        0.575879
grad_100_value   0.575879        1.000000
```



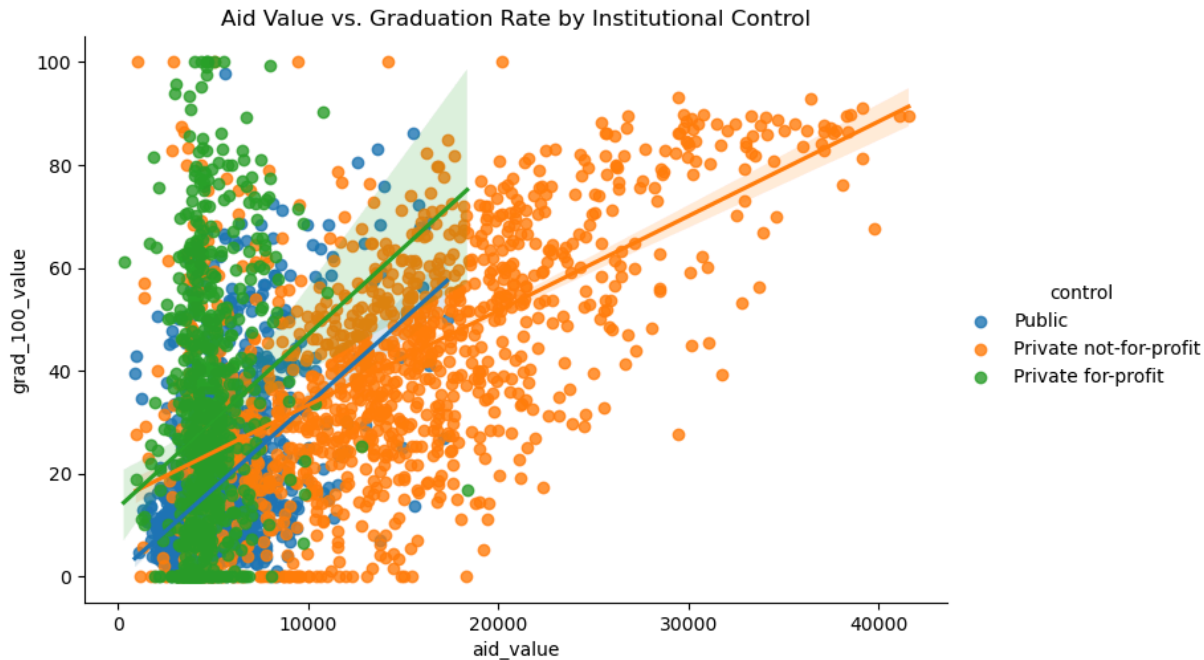
What I see?

The scatterplot shows a clear upward trend where schools providing more aid generally have higher graduation rates. And while the overall trend is positive, there is still a large cluster of schools with lower aid values (which show a much wider range of graduation rates from 0% to 100%).

In [18]:
```
import seaborn as sns

sns.lmplot(data=df_college, x='aid_value', y='grad_100_value', hue='control', aspec
plt.title('Aid Value vs. Graduation Rate by Institutional Control')
plt.show()

print("Correlation by Control Type")
print(df_college.groupby('control')[['aid_value', 'grad_100_value']].corr().iloc[0:
```

```
print("\nCovariance by Control Type")
print(df_college.groupby('control')[['aid_value', 'grad_100_value']].cov().iloc[0::
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/seaborn/axisgrid.py:118: UserWarning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```




```
Correlation by Control Type
control
Private for-profit      aid_value    0.188363
Private not-for-profit  aid_value    0.601591
Public                  aid_value    0.482481
Name: grad_100_value, dtype: float64

Covariance by Control Type
control
Private for-profit      aid_value      6897.524957
Private not-for-profit  aid_value    109274.123337
Public                  aid_value     15355.146212
Name: grad_100_value, dtype: float64
```

What I see?

All three types of schools shown an upward trend, but the steepness of the lines show that the relationship between aid and graduation is different for each group. For example, Private not-for-profit have the most spread out data between higher aid and higher graduation rates.

Which kinds of institutions does aid seem to vary positively w/ graduation rates?

Aid varies positively with graduation rates for all three types because all their correlation numbers are positive.

Q6. In class, we talked about how to compute the sample mean of a variable X,

$$m(X) = \frac{1}{N}\sum_{i=1}^{N} x_i$$

and sample covariance of two variables X and Y,

$$\text{cov}(X, Y) = \frac{1}{N}\sum_{i=1}^{N}(x_i - m(X))(y_i - m(Y))).$$

Recall, the sample variance of X is

$$s^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - m(X))^2.$$

It can be very helpful to understand some basic properties of these statistics. If you want to write your calculations on a piece of paper, take a photo, and upload that to your GitHub repo, that's probably easiest.

We're going to look at **linear transformations** of X, $Y = a + bX$. So we take each value of X, x_i, and transform it as $y_i = a + bx_i$.

1. Show that $m(a + bX) = a + b \times m(X)$.
2. Show that $\text{cov}(X, X) = s^2$.
3. Show that $\text{cov}(X, a + bY) = b \times \text{cov}(X, Y)$
4. Show that $\text{cov}(a + bX, a + bY) = b^2\text{cov}(X, Y)$. Notice, this also means that $\text{cov}(bX, bX) = b^2 s^2$.
5. Suppose $b > 0$ and let the median of X be $\text{med}(X)$. Is it true that the median of $a + bX$ is equal to $a + b \times \text{med}(X)$? Is the IQR of $a + bX$ equal to $a + b \times \text{IQR}(X)$?
6. Show by example that the means of X^2 and $\sqrt{X}$ are generally not $(m(X))^2$ and $\sqrt{m(X)}$. So, the results we derived above really depend on the linearity of the transformation $Y = a + bX$, and transformations like $Y = X^2$ or $Y = \sqrt{X}$ will not behave in a similar way.

Question 1

`![Assignment 1 Pic 1](Assignment 1 Pic 1.jpeg)`

Assignment 1 Pic 1

Question 2

Assignment 1 Pic 2

Question 3

Assignment 1 Pic 3

Question 4

Assignment 1 Pic 4

Question 5

Assignment 1 Pic 5

Question 6

Assignment 1 Pic 6

Q7. This question provides some practice doing exploratory data analysis and visualization.

We'll use the `ames_prices.csv` dataset. The "relevant" variables for this question are:

- `price` - Sale price value of the house
- `Bldg.Type` - Building type of the house (single family home, end-of-unit townhome, duplex, interior townhome, two-family conversion)

1. Load the `college_completion.csv` data with Pandas.
2. Make a kernel density plot of price and compute a describe table. Now, make a kernel density plot of price conditional on building type, and use `.groupby()` to make a describe type for each type of building. Which building types are the most expensive, on average? Which have the highest variance in transaction prices?
3. Make an ECDF plot of price, and compute the sample minimum, .25 quantile, median, .75 quantile, and sample maximum (i.e. a 5-number summary).
4. Make a boxplot of price. Are there outliers? Make a boxplot of price conditional on building type. What patterns do you see?
5. Make a dummy variable indicating that an observation is an outlier.
6. Winsorize the price variable, and compute a new kernel density plot and describe table. How do the results change?

Question 1

```
In [19]: df_college = pd.read_csv('college_completion.csv')
         df_college.head()
```

Out[19]:

	index	unitid	chronname	city	state	level	control	basic	hbcu
0	0	100654	Alabama A&M University	Normal	Alabama	4-year	Public	Masters Colleges and Universities--larger prog...	X
1	1	100663	University of Alabama at Birmingham	Birmingham	Alabama	4-year	Public	Research Universities--very high research acti...	NaN
2	2	100690	Amridge University	Montgomery	Alabama	4-year	Private not-for-profit	Baccalaureate Colleges--Arts & Sciences	NaN
3	3	100706	University of Alabama at Huntsville	Huntsville	Alabama	4-year	Public	Research Universities--very high research acti...	NaN
4	4	100724	Alabama State University	Montgomery	Alabama	4-year	Public	Masters Colleges and Universities--larger prog...	X

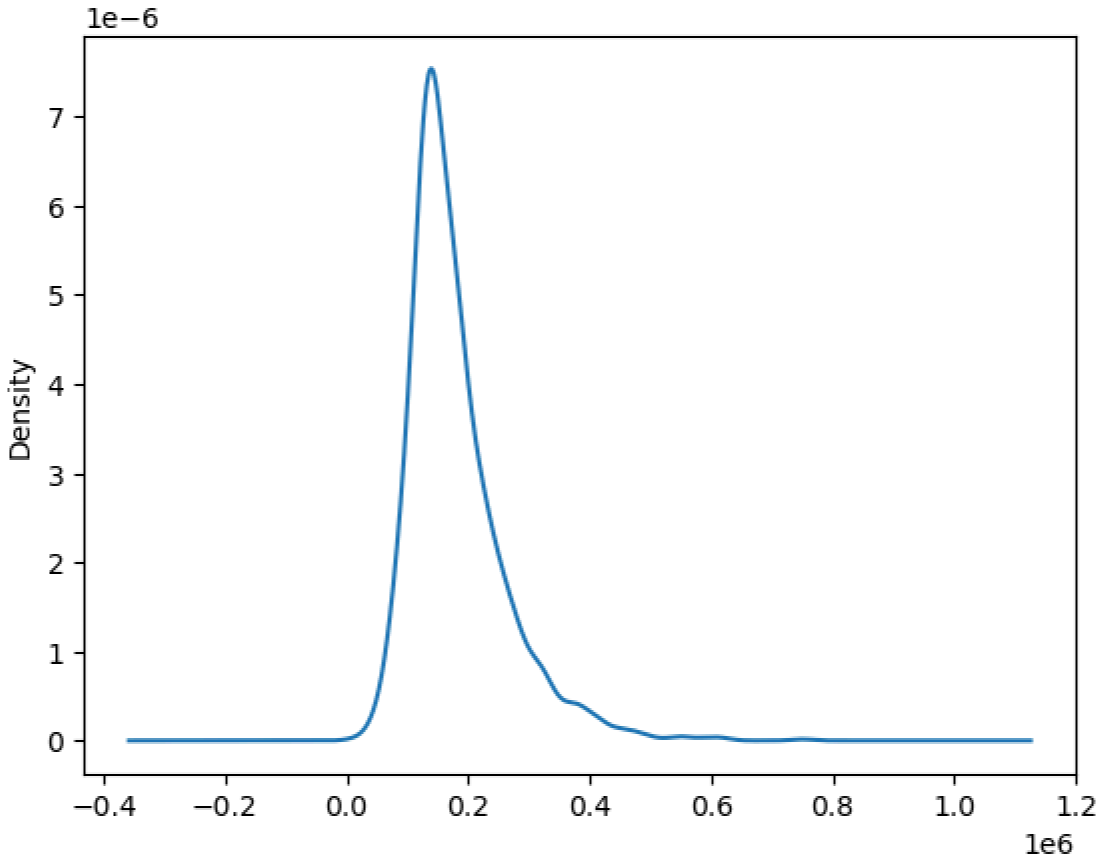
5 rows × 63 columns

Question 2

In [22]:

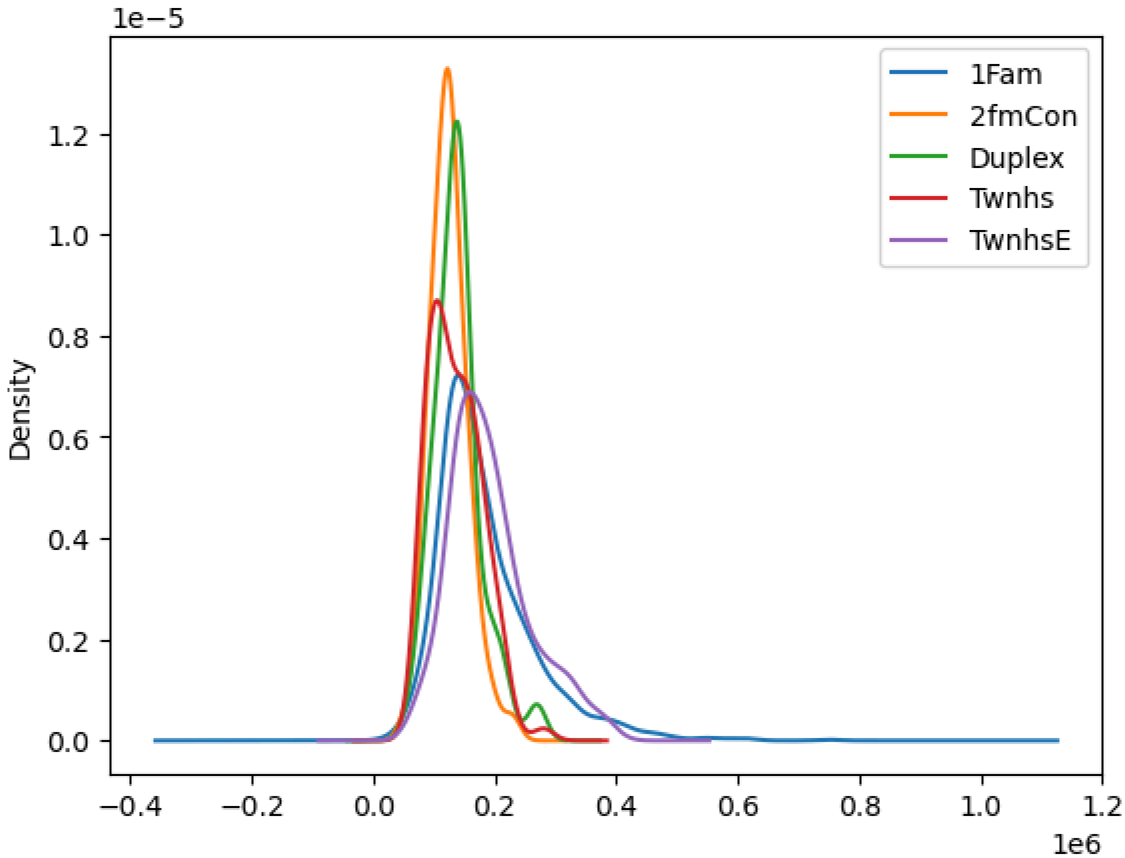
```
df_ames = pd.read_csv('ames_prices.csv')

df_ames['price'].plot(kind='density')

print(df_ames['price'].describe())
```

```
count      2930.000000
mean     180796.060068
std       79886.692357
min       12789.000000
25%      129500.000000
50%      160000.000000
75%      213500.000000
max      755000.000000
Name: price, dtype: float64
```



In [23]:
```python
df_ames.groupby('Bldg.Type')['price'].plot(kind='density', legend=True)

print(df_ames.groupby('Bldg.Type')['price'].describe())
```

```
            count           mean           std      min       25%       50%  \
Bldg.Type
1Fam       2425.0  184812.041237  82821.802329  12789.0  130000.0  165000.0
2fmCon       62.0  125581.709677  31089.239840  55000.0  106562.5  122250.0
Duplex      109.0  139808.935780  39498.973534  61500.0  118858.0  136905.0
Twnhs       101.0  135934.059406  41938.931130  73000.0  100500.0  130000.0
TwnhsE      233.0  192311.914163  66191.738021  71000.0  145000.0  180000.0

                75%       max
Bldg.Type
1Fam       220000.0  755000.0
2fmCon     140000.0  228950.0
Duplex     153337.0  269500.0
Twnhs      170000.0  280750.0
TwnhsE     222000.0  392500.0
```



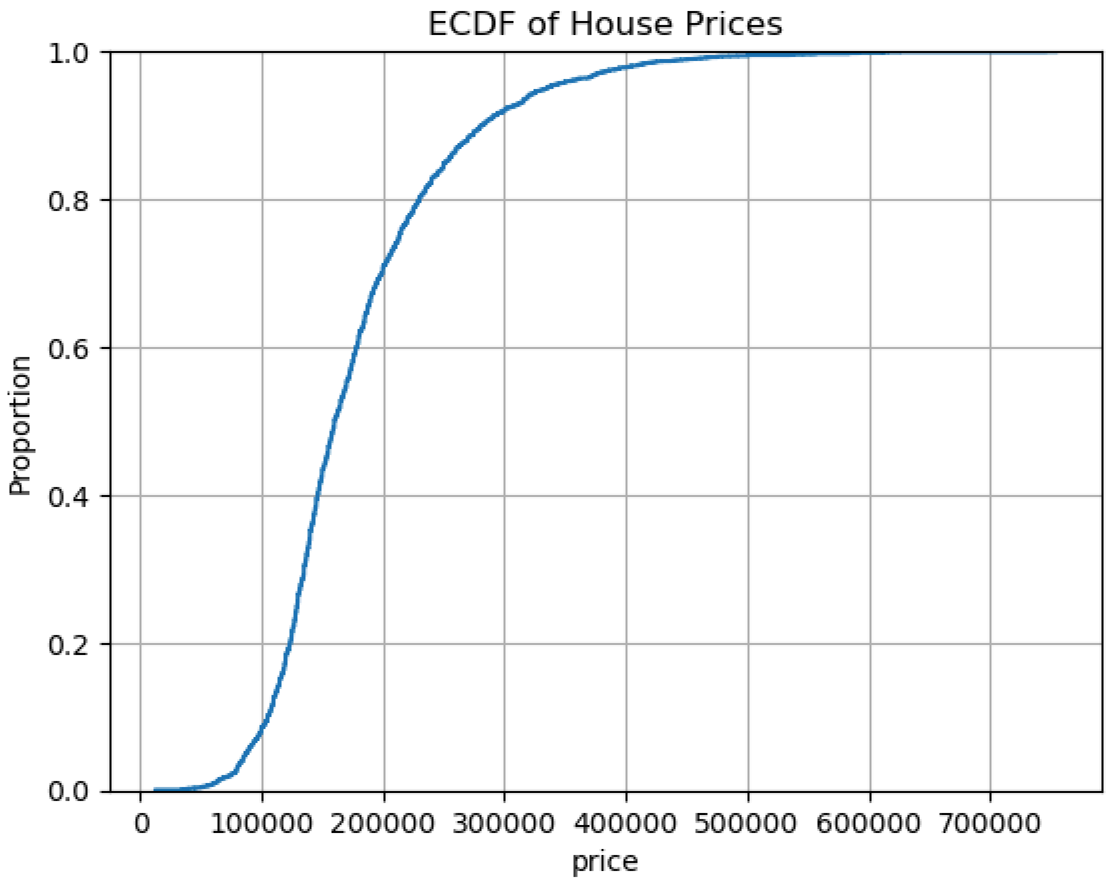
Answer to Question 2

On average, "Townhouse End Units" are the most expensive, followed by "Single Family" homes. "1Fam" homes have the highest variance in transaction prices, shown by their significantly higher standard deviation & a max price reaching $755,000.

Question 3

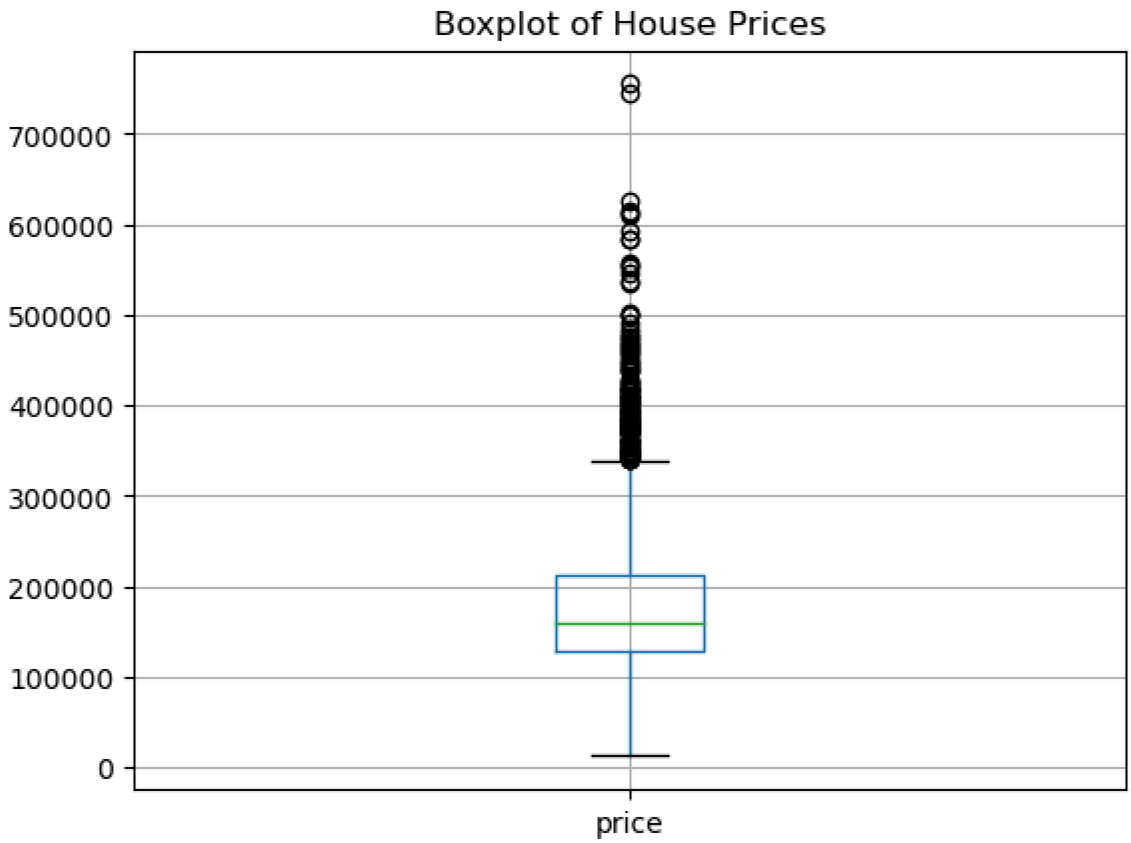
In [24]:
```
sns.ecdfplot(data=df_ames, x='price')
plt.title('ECDF of House Prices')
plt.grid(True)

five_number_summary = df_ames['price'].describe()[['min', '25%', '50%', '75%', 'max
print(five_number_summary)
```

```
min     12789.0
25%    129500.0
50%    160000.0
75%    213500.0
max    755000.0
Name: price, dtype: float64
```



Question 4

In [25]:
```
df_ames.boxplot(column='price')

plt.title('Boxplot of House Prices')
plt.show()
```

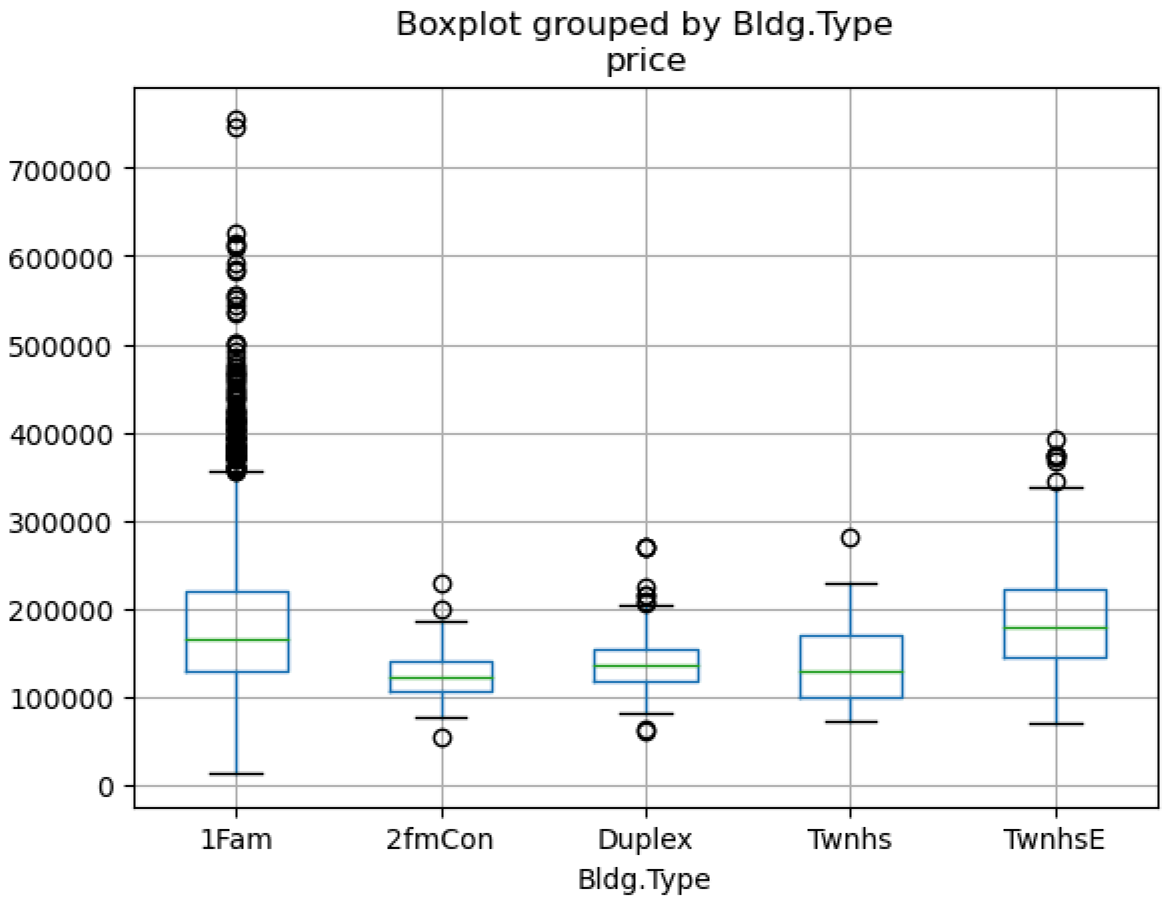


Is there any outliers?

The boxplot shows many outliers represented by the long trail of black circles above the top whisker, indicating several houses sold for much higher prices than the average.

In [28]:
```
df_ames.boxplot(column='price', by='Bldg.Type')
```

Out[28]:
```
<Axes: title={'center': 'price'}, xlabel='Bldg.Type'>
```



Pattern I see?

The "1-Fam" category shows the widest range of prices and the most extreme outliers, showing both standard and high-luxury homes. Building types like "2fmCon" and "Duplex" have much smaller boxes and fewer outliers, showing their transaction prices are more concentrated in a lower price bracket.

Question 5

In [29]:
```
Q1 = df_ames['price'].quantile(0.25)
Q3 = df_ames['price'].quantile(0.75)
IQR = Q3 - Q1

upper_limit = Q3 + 1.5 * IQR
lower_limit = Q1 - 1.5 * IQR

df_ames['is_outlier'] = ((df_ames['price'] > upper_limit) | (df_ames['price'] < low

print(df_ames['is_outlier'].value_counts())
```

```
is_outlier
0    2793
1     137
Name: count, dtype: int64
```

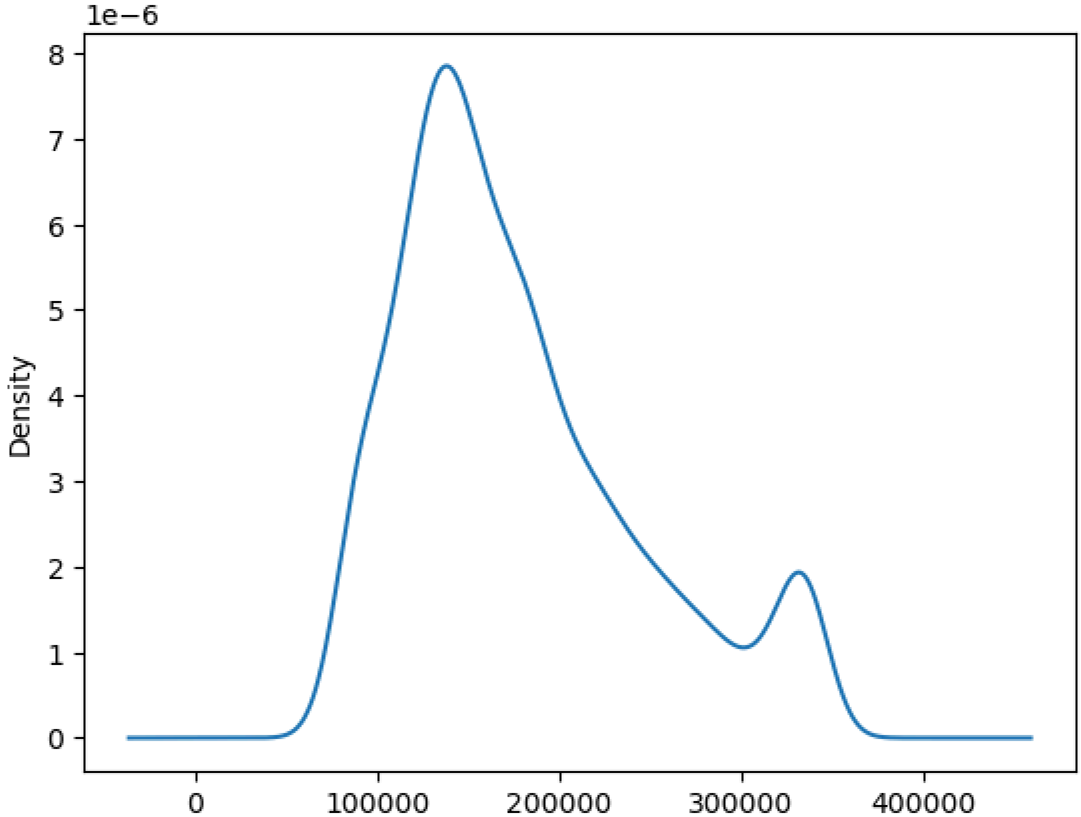
The code creates a label to separate between typical house prices and extreme ones. It calculates a "normal range" based on the middle 50% of the data and marks anything outside those boundaries as an outlier. The result at the end shows that while most houses are normal, there are 137 specific properties with unusually higher prices.

Question 6

In [31]:
```
from scipy.stats.mstats import winsorize

df_ames['price_win'] = winsorize(df_ames['price'], limits=[0.05, 0.05])

df_ames['price_win'].plot(kind='density')

print(df_ames['price_win'].describe())
```

```
count      2930.000000
mean     177632.528669
std       66195.453960
min       87500.000000
25%      129500.000000
50%      160000.000000
75%      213500.000000
max      335000.000000
Name: price_win, dtype: float64
```

```
/opt/conda/envs/anaconda-panel-2023.05-py310/lib/python3.11/site-packages/numpy/lib/
function_base.py:4737: UserWarning: Warning: 'partition' will ignore the 'mask' of t
he MaskedArray.
  arr.partition(
```



Answer to Question 6

The maximum price dropped from $755,000 to 355,000 because the high-end outliers were capped at the 95th percentile. The average price decreased as well, showing that those extreme outliers were inflating the overall market average.

#1. $m(a + bx) = {}^1/_N \sum_{i=1}^{N} (a + bx_i)$

$= {}^1/_N \left(\sum_{i=1}^{N} a + \sum_{i=1}^{N} b x_i \right) = \frac{1}{N} \left(N \times a + b \sum_{i=1}^{N} x_i \right)$

$= a + b \times \left(\frac{1}{N} \sum_{i=1}^{N} x_i \right) = a + b + m(x)$

#2. $\mathrm{cov}(X, Y) = 1/N \sum_{i=1}^{N} (x_i - m(X))(y_i - m(Y))$

$\mathrm{cov}(X, X) = 1/N \sum_{i=1}^{N} (x_i - m(x))^2$

$\mathrm{cov}(x, x) = s^2$

#3. Property of means $\rightarrow$ $m(a + bY) = a + b \times m(Y)$

$$cov(X, a + bY) = \frac{1}{N} \sum_{i=1}^{N} (x_i - m(x))\big((a + by_i) - (a + b \times m(Y))\big)$$

$$(a + by_i) - (a + b \times m(Y)) = by_i - b \times m(Y) = b(y_i - m(Y))$$

Factor $\rightarrow$ $cov(x, a + bY) = b \times \left[\frac{1}{N} \sum_{i=1}^{N} (x_i - m(x))(y_i - m(Y))\right]$

$\rightarrow$ $cov(X, a + bY) = b \times cov(x, Y)$

#4. $X' = a + bX$ and $Y' = a + bY$

$\rightarrow$ $m(X') = a + b \cdot m(X)$ and $m(Y') = a + b \cdot m(Y)$

$cov(a + bX, a + bY) = b \cdot cov(a + bX, Y)$

$\rightarrow b \cdot cov(a + bX, Y) = b \cdot [b \cdot cov(X, Y)]$

$\longrightarrow cov(a + bX, a + bY) = b^2 cov(X, Y]$

#5 Does med $(a + bX) = a + b \times \text{med}(x)$?

- when you multiply every value by b where $(b>0)$, the middle value is multiplied by b
- when you add a constant a to every value, the middle value shifts by exactly a

→ Yes it is TRUE!

Is the IQR of $a + bX$ = to $a + b \times IQR(x)$?

- Adding a constant "a" shifts all data points equally. So the distance between the spread doesn't change.
- Multiplying by "b" scales the distance between Q3 and Q1 by exactly "b".

→ No, it is NOT equal to $a + b \times IQR(x)$

#6. $m(x^2) \neq (m(x))^2$ → lets use $X = [2, 4]$

Mean of $x = 3$ → Mean squared = 9

$x^2 = 2^2, 4^2 = [4, 16]$ → $m(x^2) = 10$

→ $10 \neq 9$ so $m(x^2) \neq (m(x))^2$

$m(\sqrt{x}) \neq \sqrt{m(x)}$ → lets use $X = [1, 9]$

Mean of $x = 5$ → Square root of mean ≈ 2.25

$\sqrt{x} = \sqrt{1}, \sqrt{9} = [1, 3]$ → $m(\sqrt{x}) = 2$

$2 \neq 2.25$ so $m(\sqrt{x}) \neq \sqrt{m(x)}$