Pandas Data Frame

In [ ]:

```
'''
Q1: (1) Read "diamonds.csv" into a Pandas data frame (i.e., "diamonds_db").
    (2) Sort "diamonds_db" by "depth" and return the top 10 diamonds having the highest
    (3) Return the number of diamonds for each cut.
    (4) Return descriptive statistics of columns x, y, and z.
'''
```

In [2]:

```
import pandas as pd
import numpy as np

diamonds_db = pd.read_csv('diamonds.csv')
```

In [11]:

```
diamonds_db.head(10).sort_values('depth', ascending = False)
```

Out[11]:

	carat	cut	color	clarity	depth	table	price	x	y	z
8	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
4	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
5	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
3	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
6	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
7	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
0	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
1	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
9	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39
2	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31

In [15]:

```
cut = diamonds_db['cut']
cut.value_counts()
```

Out[15]:

```
Ideal        21551
Premium      13791
Very Good    12082
Good          4906
Fair          1610
Name: cut, dtype: int64
```

In [17]:

```
x = diamonds_db['x']
x.describe()
```

Out[17]:

```
count    53940.000000
mean         5.731157
std          1.121761
```

```
min          0.000000
25%          4.710000
50%          5.700000
75%          6.540000
max         10.740000
Name: x, dtype: float64
```

In [18]:

```
y = diamonds_db['y']
y.describe()
```

Out[18]:

```
count    53940.000000
mean         5.734526
std          1.142135
min          0.000000
25%          4.720000
50%          5.710000
75%          6.540000
max         58.900000
Name: y, dtype: float64
```

In [19]:

```
z = diamonds_db['z']
z.describe()
```

Out[19]:

```
count    53940.000000
mean         3.538734
std          0.705699
min          0.000000
25%          2.910000
50%          3.530000
75%          4.040000
max         31.800000
Name: z, dtype: float64
```

In [ ]:

```
'''
Q2: (1) Read "employee.csv" into a Pandas data frame (i.e., "employee_db").
    (2) Return how many null values are in "BASE_SALARY" column.
    (3) Change null values in the "BASE_SALARY" column to 0.
    (4) Drop any rows having null values in "employee_db".
    (5) Return a concise summary of "employee_db".
'''
```

In [3]:

```
employee_db = pd.read_csv('employee.csv')
```

In [9]:

```
employee_db['BASE_SALARY'].isnull().sum()
```

Out[9]:

```
114
```

In [10]:

```
employee_db = employee_db.fillna(0)
```

In [11]:

```
employee_db = employee_db.dropna()
```

In [12]:

```
type(employee_db)
```

Out[12]:

```
pandas.core.frame.DataFrame
```

In [13]:

```
employee_db.shape
```

Out[13]:

```
(2000, 10)
```

In [6]:

```
employee_db.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   UNIQUE_ID          2000 non-null   int64  
 1   POSITION_TITLE     2000 non-null   object 
 2   DEPARTMENT         2000 non-null   object 
 3   BASE_SALARY        1886 non-null   float64
 4   RACE               1965 non-null   object 
 5   EMPLOYMENT_TYPE    2000 non-null   object 
 6   GENDER             2000 non-null   object 
 7   EMPLOYMENT_STATUS  2000 non-null   object 
 8   HIRE_DATE          2000 non-null   object 
 9   JOB_DATE           1997 non-null   object 
dtypes: float64(1), int64(1), object(8)
memory usage: 156.4+ KB
```

In [7]:

```
employee_db.head()
```

Out[7]:

	UNIQUE_ID	POSITION_TITLE	DEPARTMENT	BASE_SALARY	RACE	EMPLOYMENT_TYPE
0	0	ASSISTANT DIRECTOR (EX LVL)	Municipal Courts Department	121862.0	Hispanic/Latino	Full Time
1	1	LIBRARY ASSISTANT	Library	26125.0	Hispanic/Latino	Full Time
2	2	POLICE OFFICER	Houston Police Department-HPD	45279.0	White	Full Time
3	3	ENGINEER/OPERATOR	Houston Fire Department (HFD)	63166.0	White	Full Time
4	4	ELECTRICIAN	General Services Department	56347.0	White	Full Time

In [8]:

```
employee_db.tail()
```

Out[8]:

	UNIQUE_ID	POSITION_TITLE	DEPARTMENT	BASE_SALARY	RACE	EMPLOYMENT_TYPE
1995	1995	POLICE OFFICER	Houston Police Department-HPD	43443.0	White	Full Time
1996	1996	COMMUNICATIONS CAPTAIN	Houston Fire Department (HFD)	66523.0	Black or African American	Full Time
1997	1997	POLICE OFFICER	Houston Police Department-HPD	43443.0	White	Full Time
1998	1998	POLICE OFFICER	Houston Police Department-HPD	55461.0	Asian/Pacific Islander	Full Time
1999	1999	FIRE FIGHTER	Houston Fire Department (HFD)	51194.0	Hispanic/Latino	Full Time

In [ ]:

```
'''
Q3: (1) Read "amzn_stock.csv" into a Pandas data frame (i.e., "amzn_stock_db").
    (2) Return the count of "Date" for when:
        1) The gap of "High" and "Low" is greater than or equal to 10; and
        2) "Volume" is lower than 5000000.
    (3) Create another data frame, "amzn_stock_db_filtered",
        which includes "Date", "Open", "Gap" (i.e., the gap of "High" and "Low"), and "V
'''
```

In [50]:

```
amzn_stock_db = pd.read_csv('amzn_stock.csv')
```

In [39]:

```
Date = ((amzn_stock_db['High'] - amzn_stock_db['Low']) >= 10) & (amzn_stock_db['Volume'
Date.filter(like = 'True').count()
```

Out[39]: 254

In [46]:

```
amzn_stock_db_filtered = Date.filter(like = 'True')
amzn_stock_db_filtered
```

Out[46]:

	Date	Open	High	Low	Close	Volume	Gap
0	1/25/2013	275.00	284.72	274.40	283.99	4974945	10.32
1	1/28/2013	283.78	284.48	274.40	276.04	4322557	10.08
2	4/18/2013	266.81	266.99	256.60	259.42	3138006	10.39
3	6/7/2013	269.74	280.10	269.13	276.87	4632539	10.97

	Date	Open	High	Low	Close	Volume	Gap
4	10/8/2013	311.50	311.54	300.27	303.23	3171592	11.27
...	...	...	...	...	...	...	...
249	7/10/2017	985.00	999.44	983.50	996.47	3546268	15.94
250	7/11/2017	993.00	995.99	983.72	994.13	2982726	12.27
251	7/12/2017	1000.65	1008.55	998.10	1006.51	3608574	10.45
252	7/13/2017	1004.62	1006.88	995.90	1000.63	2880769	10.98
253	7/17/2017	1004.69	1014.75	1003.81	1010.04	3668721	10.94

254 rows × 7 columns

In [ ]:

```
'''
Q4: (1) Read "aapl_stock.csv" into a Pandas data frame (i.e., "aapl_stock_db").
    (2) Create another data frame, "stock_db", by concatenating "aapl_stock_db" to "amz
        - To distinguish amzn and aapl, use keys (i.e., Amazon, Apple) as a name (i.e.,
    (3) Return the "Close" and "Volume" of Apple on 2017-05-15.
```

In [64]:

```
aapl_stock_db = pd.read_csv('aapl_stock.csv')
```

In [66]:

```
stock_db = pd.concat([amzn_stock_db, aapl_stock_db], keys = ['Amazon', 'Apple'])
stock_db
```

Out[66]:

		Date	Open	High	Low	Close	Volume
Amazon	**0**	1/4/2010	136.25	136.61	133.14	133.90	7600543
	1	1/5/2010	133.43	135.48	131.81	134.69	8856456
	2	1/6/2010	134.60	134.73	131.65	132.25	7180977
	3	1/7/2010	132.01	132.32	128.80	130.00	11030124
	4	1/8/2010	130.56	133.68	129.03	133.52	9833829
...	**...**	...	...	...	...	...	...
Apple	**1670**	7/10/2017	144.11	145.95	143.37	145.06	21090636
	1671	7/11/2017	144.73	145.85	144.38	145.53	19781836
	1672	7/12/2017	145.87	146.18	144.82	145.74	24884478
	1673	7/13/2017	145.50	148.49	145.44	147.77	25199373
	1674	7/14/2017	147.97	149.33	147.33	149.04	20132061

3571 rows × 6 columns

In [63]:

```
stock_db[['Date', 'Close', 'Volume']].loc['Apple'].filter(like = '2017-05-15')
```

Out[63]:

	Date	Close	Volume
0	5/15/2017	155.7	26009719

In [ ]:

```
'''
Q5: (1) Read "food_prices.csv" into a Pandas data frame (i.e., "food_prices_db").
        - Drop the last row: the price of steak at Store B in 2015.
    (2) Read "food_transactions.csv" into a Pandas data frame (i.e., "food_transactions_
    (3) Create a new data frame, "customer_purchase" with following:
        1) Columns: "custid", "item", "store", "quantity", "price", "Date", and "purcha
        2) Note: "purchase" refers how much a customer spent to buy an item at a store
        3) If there is a row having any null values in the data frame, "customer_purcha
'''
```

In [75]:

```
food_prices_db = pd.read_csv('food_prices.csv')
food_prices_db = food_prices_db.drop([8])
```

In [77]:

```
food_transactions_db = pd.read_csv('food_transactions.csv')
```

In [90]:

```
customer_purchase = pd.merge(food_prices_db, food_transactions_db, how = 'outer', on =
customer_purchase['purchase'] = customer_purchase['quantity'] * customer_purchase['pric
customer_purchase = customer_purchase[['custid', 'item', 'store', 'quantity', 'price',

customer_purchase = customer_purchase.dropna()
customer_purchase = customer_purchase.reset_index()
customer_purchase
```

Out[90]:

	index	custid	item	store	quantity	price	Date	purchase
0	0	1.0	pear	A	5.0	0.99	2017.0	4.95
1	1	2.0	pear	B	1.0	1.99	2017.0	1.99
2	3	2.0	peach	B	2.0	3.49	2017.0	6.98
3	4	1.0	banana	A	10.0	0.39	2017.0	3.90
4	7	2.0	steak	B	3.0	6.99	2017.0	20.97
5	8	2.0	steak	B	1.0	6.99	2017.0	6.99

In [ ]:

```
'''
Q6: (1) Read "diamonds.csv" into a Pandas data frame (i.e., "diamonds_db").
    (2) Visualize the following in a single figure:
        1) The size of the figure is (20, 10);
        2) The subtitle of the figure is "Diamonds: Univariate Analysis". Its size is 2
        3) Visualize the following four subplots (2 x 2):
           - Plot_1: Bar plot of "color". Title is "Color". Rotate the xticks. The colo
           - Plot_2: Bar plot of "cut". The title is "Cut". Rotate the xticks. The colo
           - Plot_3: Box plot of "price". The title is "Price"; and
           - Plot_4: Horizontal bar plot of "clarity". The title is "Clarity". The colo
'''
```

In [3]:

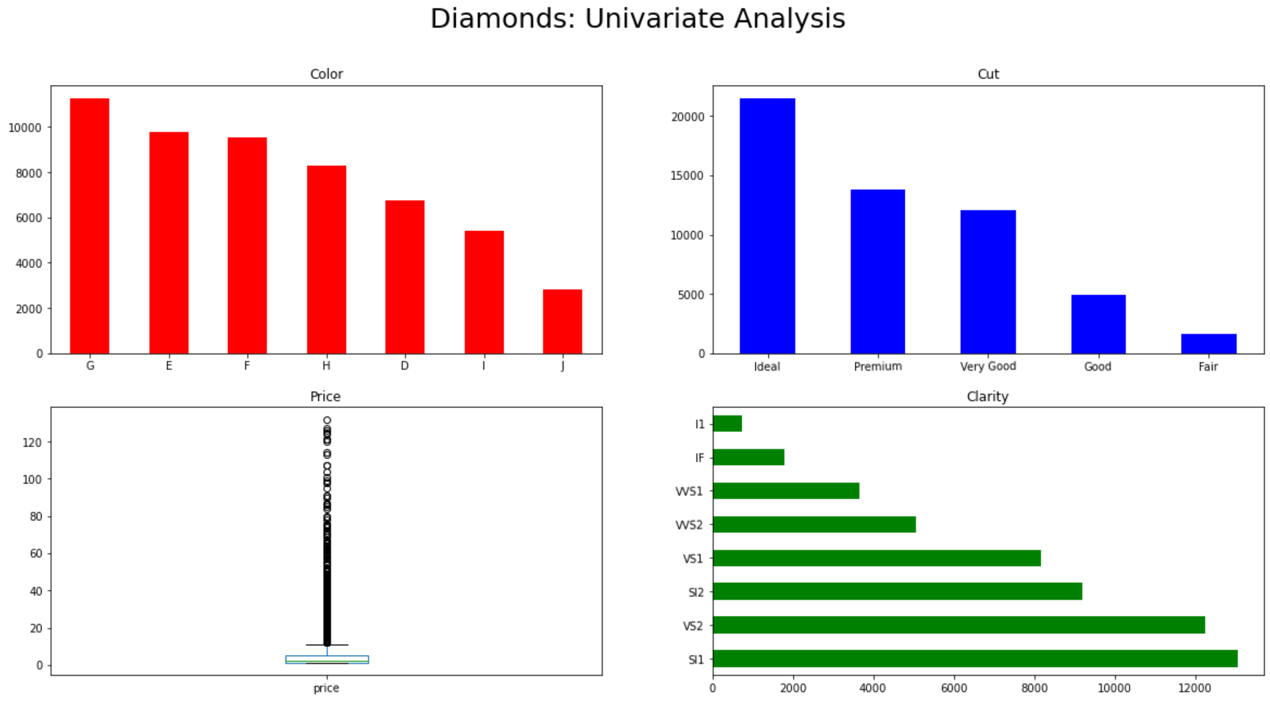
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

%matplotlib inline

diamonds_db = pd.read_csv('diamonds.csv')
```

In [4]:

```python
fig, ((ax1, ax2), (ax3, ax4)) = plt.subplots(2, 2, figsize = (20, 10))

fig.suptitle('Diamonds: Univariate Analysis', size = 25)

Plot_1 = diamonds_db['color'].value_counts()
Plot_1.plot(kind = 'bar', ax = ax1, rot = True, color = 'red', title = 'Color')

Plot_2 = diamonds_db['cut'].value_counts()
Plot_2.plot(kind = 'bar', ax = ax2, rot = True, color = 'blue', title = 'Cut')

Plot_3 = diamonds_db['price'].value_counts()
Plot_3.plot(kind = 'box', ax = ax3, rot = True, title = 'Price')

Plot_4 = diamonds_db['clarity'].value_counts()
Plot_4.plot(kind = 'barh', ax = ax4, rot = True, color = 'green', title = 'Clarity')
```

Out[4]:

```
<AxesSubplot:title={'center':'Clarity'}>
```


Diamonds: Univariate Analysis
Color
G
E
F
H
D
I
J
Cut
Ideal
Premium
Very Good
Good
Fair
Price
price
Clarity
I1
IF
VVS1
VVS2
VS1
SI2
VS2
SI1