In [1]:

```
#Part 1: How to load data file(s) using Pandas?
```

In [2]:

```
import pandas as pd
import numpy
import scipy
```

In [3]:

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

In [4]:

```
df
```

Out[4]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size	Gradu
0	1	Male	19	15000	39	Healthcare	1	4	
1	2	Male	21	35000	81	Engineer	3	3	
2	3	Female	20	86000	6	Engineer	1	1	
3	4	Female	23	59000	77	Lawyer	0	2	
4	5	Female	31	38000	40	Entertainment	2	6	
...	...	...	...	...	...	...	...	...	
1995	1996	Female	71	184387	40	Artist	8	7	
1996	1997	Female	91	73158	32	Doctor	7	7	
1997	1998	Male	87	90961	14	Healthcare	9	2	
1998	1999	Male	77	182109	4	Executive	7	2	
1999	2000	Male	90	110610	52	Entertainment	5	2	

2000 rows × 9 columns

In [5]:

```
backup_df = df
```

In [6]:

```
#Part 2: How to convert a variable to a different data type?
```

In [10]:

```
#Convert numeric variables to string variables and vice versa

numeric_input = 10
string_input = "12"

string_outcome = str(numeric_input) #Converts numeric_input to string_outcome
integer_outcome = int(string_input) #Converts string_input to integer_outcome
float_outcome = float(string_input) #Converts string_input to integer_outcome

print(string_outcome ,"Type is:", type(string_outcome))
print(integer_outcome,"Type is:", type(integer_outcome))
print(float_outcome,"Type is:", type(float_outcome))
```

```
10 Type is: <class 'str'>
12 Type is: <class 'int'>
12.0 Type is: <class 'float'>
```

In [17]:

```
#Convert character date to Date:

from datetime import datetime
char_date = 'April 1 2015 1:20 PM' #creating example character date

# %B    Month name, long version
# %b    Month name, short version
# %d    Day of month 01-31
# %Y    Year, full version
# %I    Hour 00-12
# %M    Minute 00-59
# %p    AM/PM

date_obj = datetime.strptime(char_date, '%B %d %Y %I:%M %p')
print(date_obj)
```

```
2015-04-01 13:20:00
```

In [15]:

```
#Part 3: How to transpose a Data set or dataframe using Pandas?
```

In [16]:

```
#Transposing Pandas dataframe by a variable
result= df.pivot( columns='Profession', values='Annual Income ($)')
result
```

Out[16]:

Profession	NaN	Artist	Doctor	Engineer	Entertainment	Executive	Healthcare	Homema
0	NaN	NaN	NaN	NaN	NaN	NaN	15000.0	N
1	NaN	NaN	NaN	35000.0	NaN	NaN	NaN	N
2	NaN	NaN	NaN	86000.0	NaN	NaN	NaN	N
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
4	NaN	NaN	NaN	NaN	38000.0	NaN	NaN	N
...	...	...	...	...	...	...	...	
1995	NaN	184387.0	NaN	NaN	NaN	NaN	NaN	N
1996	NaN	NaN	73158.0	NaN	NaN	NaN	NaN	N
1997	NaN	NaN	NaN	NaN	NaN	NaN	90961.0	N
1998	NaN	NaN	NaN	NaN	NaN	182109.0	NaN	N
1999	NaN	NaN	NaN	NaN	110610.0	NaN	NaN	N

2000 rows × 10 columns

In [17]:

```
#Part 4: How to sort a Pandas DataFrame?
```

In [21]:

```
#Sorting Pandas Dataframe
df.sort_values(['Annual Income ($)','Age'], ascending=[True, False])
```

Out[21]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size	Gradu
169	170	Male	32	0	63	Artist	2	2	
246	247	Male	23	0	96	Doctor	1	3	
272	273	Female	96	1000	76	Entertainment	0	3	
96	97	Female	47	2000	47	Artist	0	1	
113	114	Male	19	2000	46	Artist	1	1	
...	...	...	...	...	...	...	...	...	
638	639	Female	59	189672	8	Doctor	0	7	
1576	1577	Female	16	189689	37	Healthcare	8	5	
1825	1826	Male	7	189709	18	Artist	6	6	
1257	1258	Male	60	189945	20	Engineer	0	5	
569	570	Female	91	189974	37	Engineer	8	1	

2000 rows × 9 columns

In [26]:

```
df.sort_values(['Age'], ascending=True)
```

Out[26]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size	Gradu
443	444	Female	0	68761	16	Lawyer	1	4	
466	467	Male	0	186002	80	Doctor	15	2	
1271	1272	Female	0	61228	81	Entertainment	1	6	
1248	1249	Female	0	143082	63	Entertainment	0	7	
1583	1584	Female	0	120899	7	Marketing	2	6	
...	...	...	...	...	...	...	...	...	
1629	1630	Male	99	162762	52	Healthcare	1	1	
1188	1189	Female	99	122548	14	NaN	2	5	
351	352	Male	99	173394	4	Engineer	13	1	
361	362	Male	99	63364	61	Entertainment	1	2	
347	348	Female	99	184426	41	Artist	9	1	

2000 rows × 9 columns

In [21]:

```
#Part 5: How to create plots (Histogram, Scatter, Box Plot)?
```

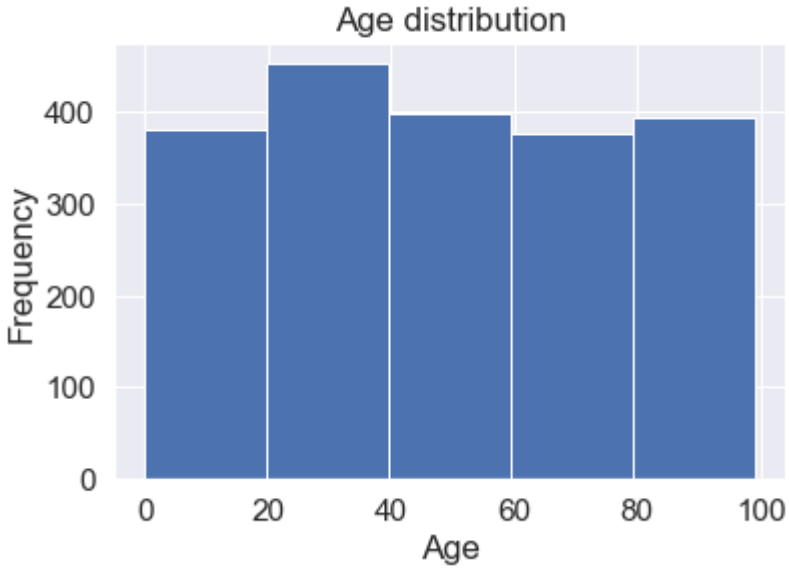
In [30]:

```
#Plot Histogram
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns

#Plots in matplotlib reside within a figure object, use plt.figure to create new figure
fig=plt.figure()

#Create one or more subplots using add_subplot, because you can't create blank figure
ax = fig.add_subplot(1,1,1)

#Variable
ax.hist(df['Age'],bins = 5)
#Labels and Title
plt.title('Age distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

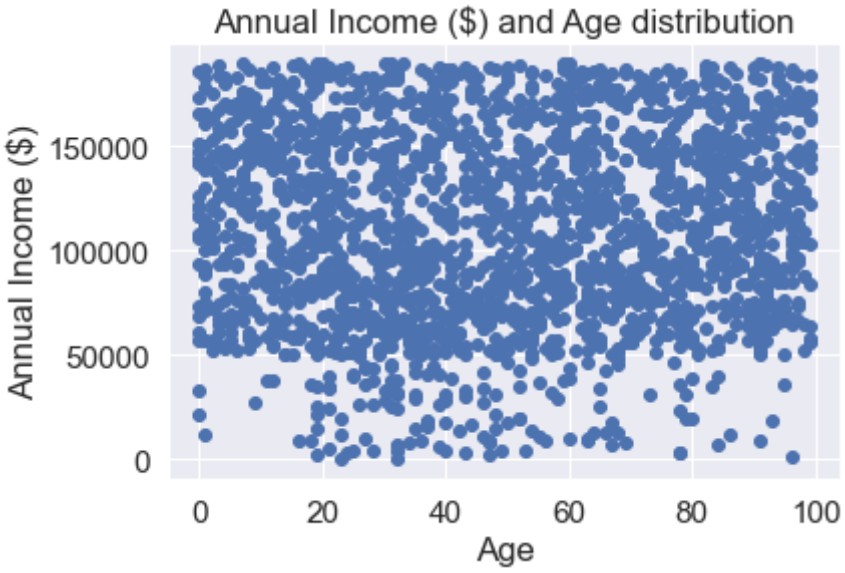


In [ ]:

```
#Scatter plot:
```

In [32]:

```
#Plots in matplotlib reside within a figure object, use plt.figure to create new figure
fig=plt.figure()
#Create one or more subplots using add_subplot, because you can't create blank figure
ax = fig.add_subplot(1,1,1)
#Variable
ax.scatter(df['Age'],df['Annual Income ($)'])
#Labels and Title
plt.title('Annual Income ($) and Age distribution')
plt.xlabel('Age')
plt.ylabel('Annual Income ($)')
plt.show()
```



In [33]:

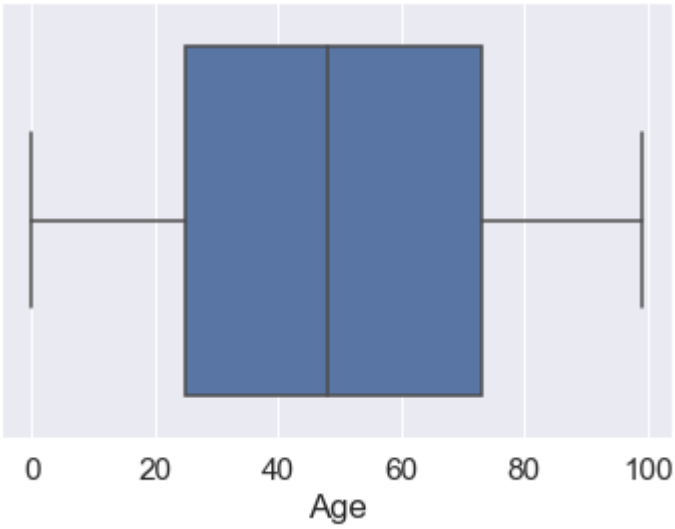
```
#Box-plot:
```

In [43]:

```
import seaborn as sns
sns.boxplot(x=df['Age'])
```

Out[43]:

```
<AxesSubplot:xlabel='Age'>
```



In [44]:

```
#Part 6: How to generate frequency tables with Pandas?
```

In [30]:

```
import pandas as pd

df=pd.read_csv("Customer.csv")
test= df.groupby(['Gender','Profession'])
test.size()
```

Out[30]:

```
Gender  Profession
Female  Artist           380
        Doctor            89
        Engineer         103
        Entertainment    133
        Executive         87
        Healthcare       196
        Homemaker         39
        Lawyer            86
        Marketing         53
Male    Artist           232
        Doctor            72
        Engineer          76
        Entertainment    101
        Executive         66
        Healthcare       143
        Homemaker         21
        Lawyer            56
        Marketing         32
dtype: int64
```

In [48]:

```
#Part 7: How to do sample Data set in Python?
```

In [38]:

```
#Create Sample dataframe
import numpy as np
import pandas as pd
from random import sample
dfr = df.sample(n=10)
#print(dfr)
dfr.head(10)
```

Out[38]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size	Gradu
482	483	Male	76	155622	81	Healthcare	0	2	
164	165	Male	50	47000	26	Doctor	0	2	
1064	1065	Male	23	122712	21	Marketing	1	1	
1764	1765	Female	63	176783	40	Doctor	9	4	
90	91	Female	68	46000	55	Entertainment	0	3	
187	188	Male	28	44000	68	Doctor	5	3	
332	333	Male	84	137641	67	Healthcare	0	5	
605	606	Female	3	155864	31	Healthcare	13	3	
1220	1221	Male	68	170957	6	Lawyer	1	3	
1750	1751	Male	79	95839	30	Executive	7	2	

In [ ]:

```
#Part 8: How to remove duplicate values of a variable in a Pandas Dataframe?
```

In [53]:

```
# Before Removing any duplicates
df.shape
```

Out[53]:

```
(2000, 9)
```

In [45]:

```
#Remove Duplicate Values based on values of variables "Gender" and "Profession"
rem_dup=df.drop_duplicates(['Gender', 'Profession'])
rem_dup.head(20)
#rem_dup.shape
```

Out[45]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size	Gradua
0	1	Male	19	15000	39	Healthcare	1	4	
1	2	Male	21	35000	81	Engineer	3	3	
2	3	Female	20	86000	6	Engineer	1	1	
3	4	Female	23	59000	77	Lawyer	0	2	
4	5	Female	31	38000	40	Entertainment	2	6	
5	6	Female	22	58000	76	Artist	0	2	
6	7	Female	35	31000	6	Healthcare	1	3	
12	13	Female	58	80000	15	Executive	0	5	
14	15	Male	37	19000	13	Doctor	0	1	
16	17	Female	35	29000	35	Homemaker	9	5	
18	19	Male	52	20000	29	Entertainment	1	4	
20	21	Male	35	96000	35	Homemaker	12	1	
23	24	Male	31	71000	73	Artist	5	2	
31	32	Female	21	34000	73	Doctor	1	2	
54	55	Female	50	18000	45	Marketing	1	1	
59	60	Male	53	89000	46	Lawyer	1	6	
79	80	Female	49	98000	42	NaN	1	1	
103	104	Male	26	52000	55	Executive	0	5	
138	139	Male	19	22000	10	Marketing	8	4	
237	238	Male	95	36000	35	NaN	0	4	

In [55]:

```
#Part 9: How to group variables in Pandas to calculate count, average, sum?
```

In [56]:

```
test = df.groupby(['Profession'])
test.describe()
```

Out[56]:

	CustomerID								
	count	mean	std	min	25%	50%	75%	max	count
Profession									
Artist	612.0	994.034314	585.230026	6.0	473.50	1007.0	1484.25	1996.0	612.0
Doctor	161.0	1010.571429	582.629135	15.0	505.00	1052.0	1493.00	1997.0	161.0
Engineer	179.0	935.692737	583.085006	2.0	372.50	916.0	1453.00	1995.0	179.0
Entertainment	234.0	1061.166667	544.322702	5.0	641.50	1048.0	1541.00	2000.0	234.0
Executive	153.0	1072.254902	594.153882	13.0	596.00	1117.0	1611.00	1999.0	153.0
Healthcare	339.0	970.876106	562.578312	1.0	501.50	950.0	1436.50	1998.0	339.0
Homemaker	60.0	1022.916667	622.422721	17.0	482.25	975.5	1640.50	1967.0	60.0
Lawyer	142.0	957.401408	595.812813	4.0	412.75	944.5	1440.75	1985.0	142.0
Marketing	85.0	1038.211765	570.284684	55.0	585.00	947.0	1561.00	1993.0	85.0

9 rows × 48 columns

In [57]:

```
#Part 10: How to recognize and Treat missing values and outliers in Pandas?
```

In [59]:

```
# Identify missing values of dataframe
df.isnull().sum()
```

Out[59]:

```
CustomerID                0
Gender                    0
Age                       0
Annual Income ($)         0
Spending Score (1-100)    0
Profession               35
Work Experience           0
Family Size               0
Graduated                17
dtype: int64
```

In [60]:

```
#Part 11: How to merge / join data sets and Pandas dataframes?
```

In [48]:

```
df_new = pd.merge(df, dfr, how = 'inner', left_index = True, right_index = True)
df_new
```

Out[48]:

	CustomerID_x	Gender_x	Age_x	Annual Income ($)_x	Spending Score (1-100)_x	Profession_x	Work Experience_x	Famil Size_
90	91	Female	68	46000	55	Entertainment	0	
164	165	Male	50	47000	26	Doctor	0	
187	188	Male	28	44000	68	Doctor	5	
332	333	Male	84	137641	67	Healthcare	0	
482	483	Male	76	155622	81	Healthcare	0	
605	606	Female	3	155864	31	Healthcare	13	
1064	1065	Male	23	122712	21	Marketing	1	
1220	1221	Male	68	170957	6	Lawyer	1	
1750	1751	Male	79	95839	30	Executive	7	
1764	1765	Female	63	176783	40	Doctor	9	

In [49]:

```
df_new.shape
```

Out[49]:

```
(10, 18)
```

In [ ]:

```

```