In [ ]:

Dataset:

Shop Customer Data is a detailed analysis of a imaginative shop's ideal customers. It helps a business to better understand its customers. The owner of a shop gets information about Customers through membership cards.

Dataset consists of 2000 records and 8 columns:

- Customer ID
- Gender
- Age
- Annual Income
- Spending Score - Score assigned by the shop, based on customer behavior and spending nature
- Profession
- Work Experience - in years
- Family Size

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

In [2]:
```
df = pd.read_csv(r'C:\Users\Mukul\Downloads\Customers.csv')
```

In [3]:
```
#check the top five data
df.head()
```

Out[3]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
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

In [4]:
```
# check last five data

df.tail()
```

Out[4]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
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

In [5]:
```
# check the columns
df.columns
```

Out[5]:
```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income ($)',
       'Spending Score (1-100)', 'Profession', 'Work Experience',
       'Family Size'],
      dtype='object')
```

In [6]:
```
# check the shape

df.shape
```

Out[6]:
```
(2000, 8)
```

In [ ]:
```

```

In [7]:
```
# check the info()

df.info()
```

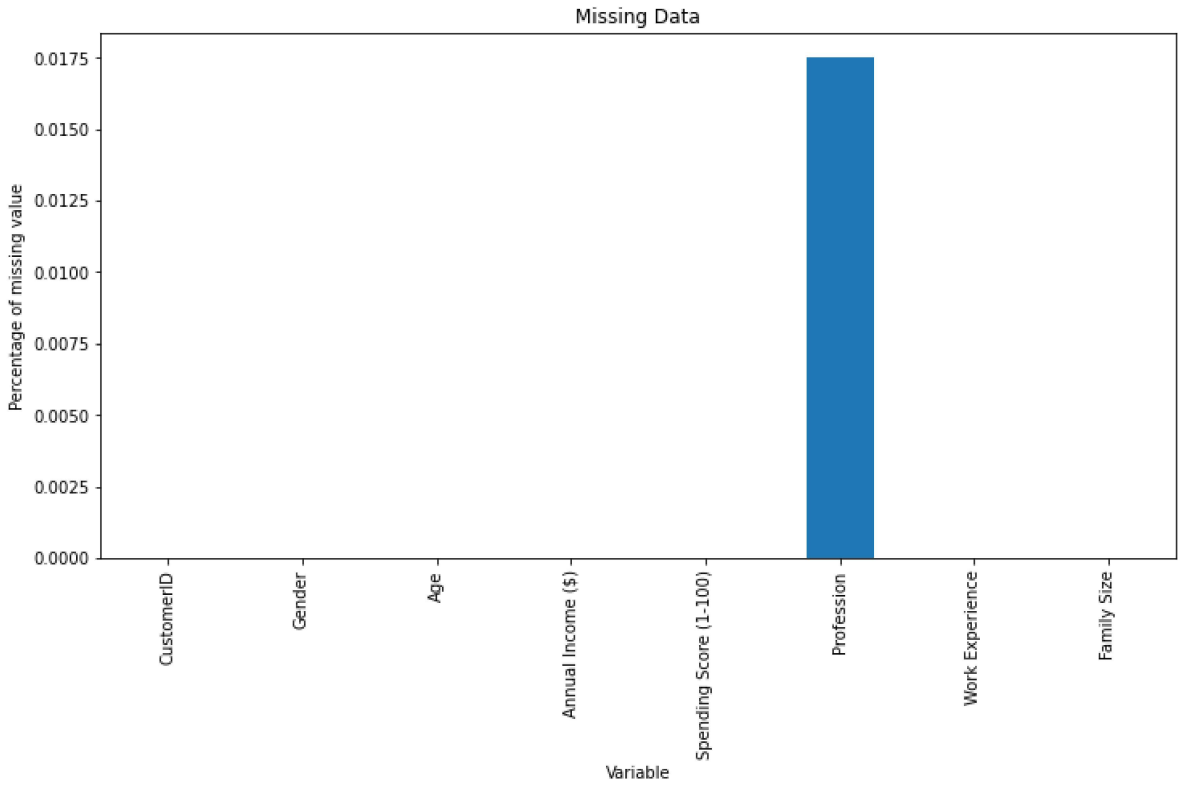
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2000 entries, 0 to 1999
Data columns (total 8 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   CustomerID              2000 non-null   int64
 1   Gender                  2000 non-null   object
 2   Age                     2000 non-null   int64
 3   Annual Income ($)       2000 non-null   int64
 4   Spending Score (1-100)  2000 non-null   int64
 5   Profession              1965 non-null   object
 6   Work Experience         2000 non-null   int64
 7   Family Size             2000 non-null   int64
dtypes: int64(6), object(2)
memory usage: 125.1+ KB
```

In [8]:
```
# check the nulll values
df.isnull().sum()
```

Out[8]:
```
CustomerID                 0
Gender                     0
Age                        0
Annual Income ($)          0
Spending Score (1-100)     0
Profession                35
Work Experience            0
Family Size                0
dtype: int64
```

In [9]:
```
# indicated the null values
df.isnull().mean().plot.bar(figsize=(12,6))
plt.xlabel("Variable")
plt.ylabel("Percentage of missing value")
plt.title('Missing Data')
```

Out[9]:
```
Text(0.5, 1.0, 'Missing Data')
```



In [10]:
```
# check the missing values of %
df.isnull().mean()*100
```

Out[10]:
```
CustomerID                0.00
Gender                    0.00
Age                       0.00
Annual Income ($)         0.00
Spending Score (1-100)    0.00
Profession                1.75
Work Experience           0.00
Family Size               0.00
dtype: float64
```

In [11]:
```
# Here are indicated the one variable is missing values : Profession
```

In [12]:
```
df['Profession'].dtypes
```

Out[12]:
```
dtype('O')
```

In [13]:
```
df['Profession'].unique()
```

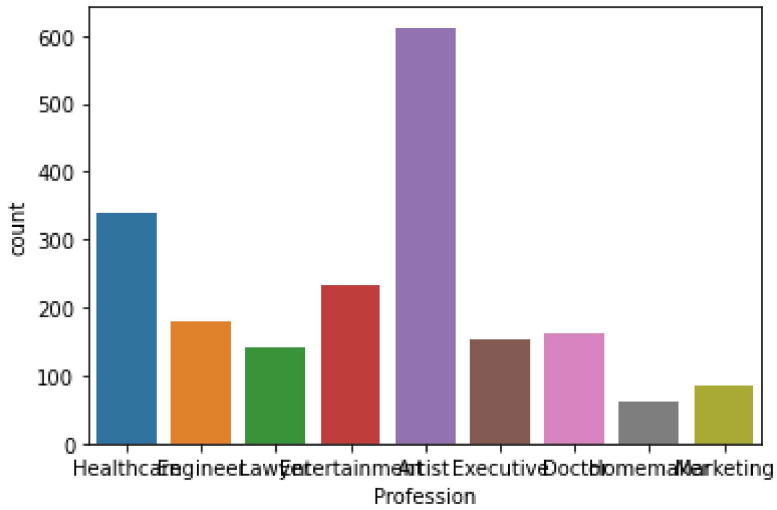
Out[13]:
```
array(['Healthcare', 'Engineer', 'Lawyer', 'Entertainment', 'Artist',
       'Executive', 'Doctor', 'Homemaker', 'Marketing', nan], dtype=object)
```

In [14]:
```
df['Profession'].value_counts()
```

Out[14]:
```
Artist           612
Healthcare       339
Entertainment    234
Engineer         179
Doctor           161
Executive        153
Lawyer           142
Marketing         85
Homemaker         60
Name: Profession, dtype: int64
```

In [15]:
```
sns.countplot(df['Profession'])
```

Out[15]:
```
<AxesSubplot:xlabel='Profession', ylabel='count'>
```



In [16]:
```
# filling the missing values

df['Profession'] = df['Profession'].fillna(df['Profession'].mode()[0])
```

In [17]:
```
df.isnull().sum()
```

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

In [18]:
```
# check the unique values

df.nunique()
```

Out[18]:
```
CustomerID                2000
Gender                       2
Age                        100
Annual Income ($)         1786
Spending Score (1-100)     101
Profession                   9
Work Experience             18
Family Size                  9
dtype: int64
```

In [19]:
```
# check duplicated values
df.duplicated().sum()
```

Out[19]:
```
0
```

In [20]:
```
# check total_counts
df.count()
```

Out[20]:
```
CustomerID                2000
Gender                    2000
Age                       2000
Annual Income ($)         2000
Spending Score (1-100)    2000
Profession                2000
Work Experience           2000
Family Size               2000
dtype: int64
```

Segregated the Numerical and Categorical Values

In [21]:
```
[feature for feature in df.columns]
```

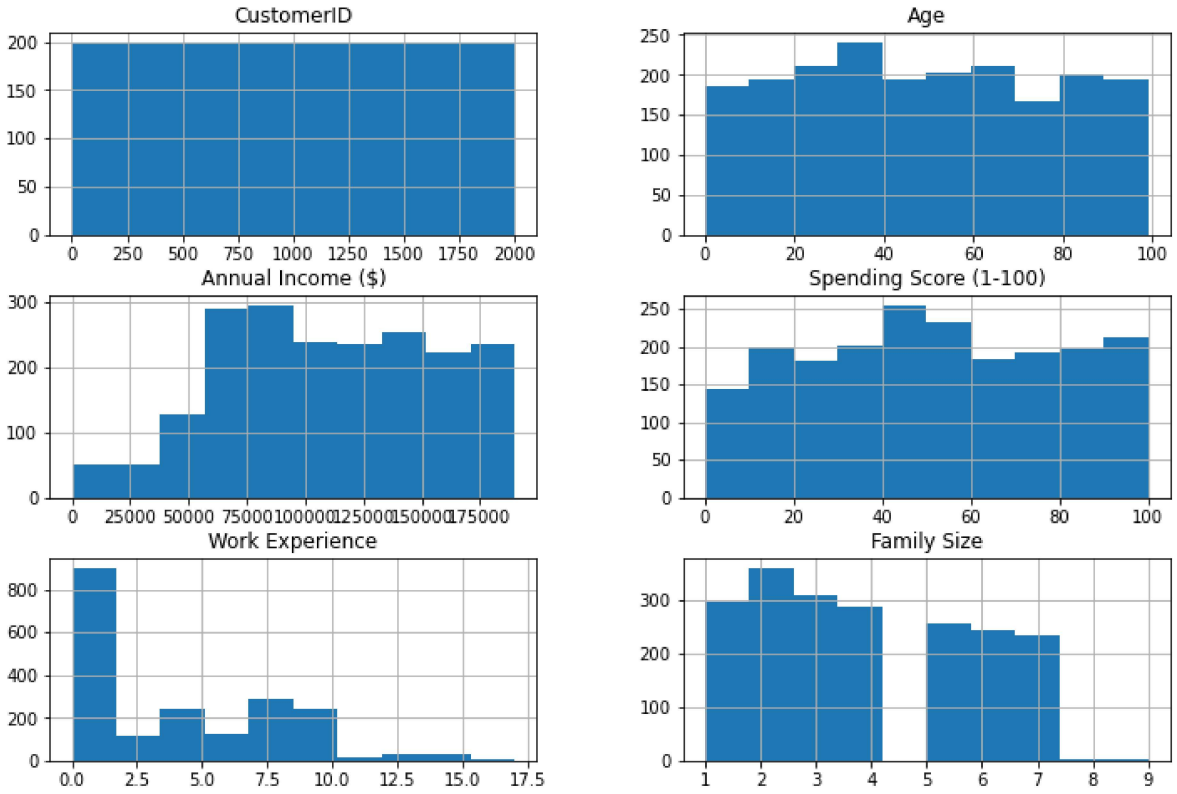
Out[21]:
```
['CustomerID',
 'Gender',
 'Age',
 'Annual Income ($)',
 'Spending Score (1-100)',
 'Profession',
 'Work Experience',
 'Family Size']
```

In [22]:
```
# numerical Columns

num_col=[feature for feature in df.columns if df[feature].dtypes!='object']
num_col
```

Out[22]:
```
['CustomerID',
 'Age',
 'Annual Income ($)',
 'Spending Score (1-100)',
 'Work Experience',
 'Family Size']
```

In [23]:
```
# indicated the numerical columns
df.hist(figsize=(12,8))
```

Out[23]:
```
array([[<AxesSubplot:title={'center':'CustomerID'}>,
        <AxesSubplot:title={'center':'Age'}>],
       [<AxesSubplot:title={'center':'Annual Income ($)'}>,
        <AxesSubplot:title={'center':'Spending Score (1-100)'}>],
       [<AxesSubplot:title={'center':'Work Experience'}>,
        <AxesSubplot:title={'center':'Family Size'}>]], dtype=object)
```

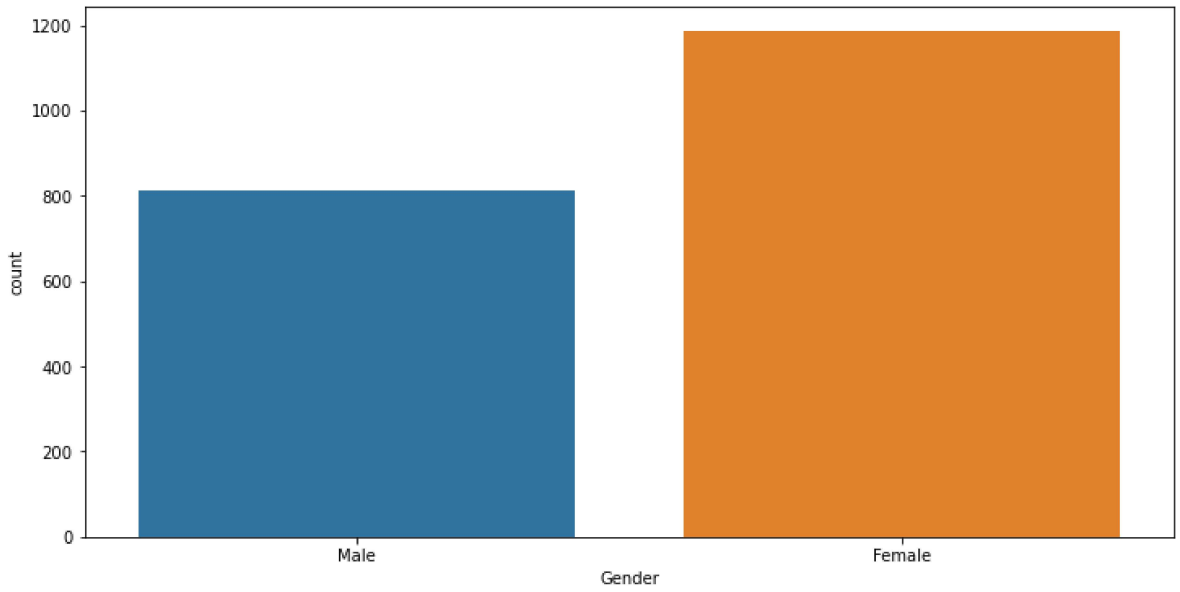


In [ ]:
```
```

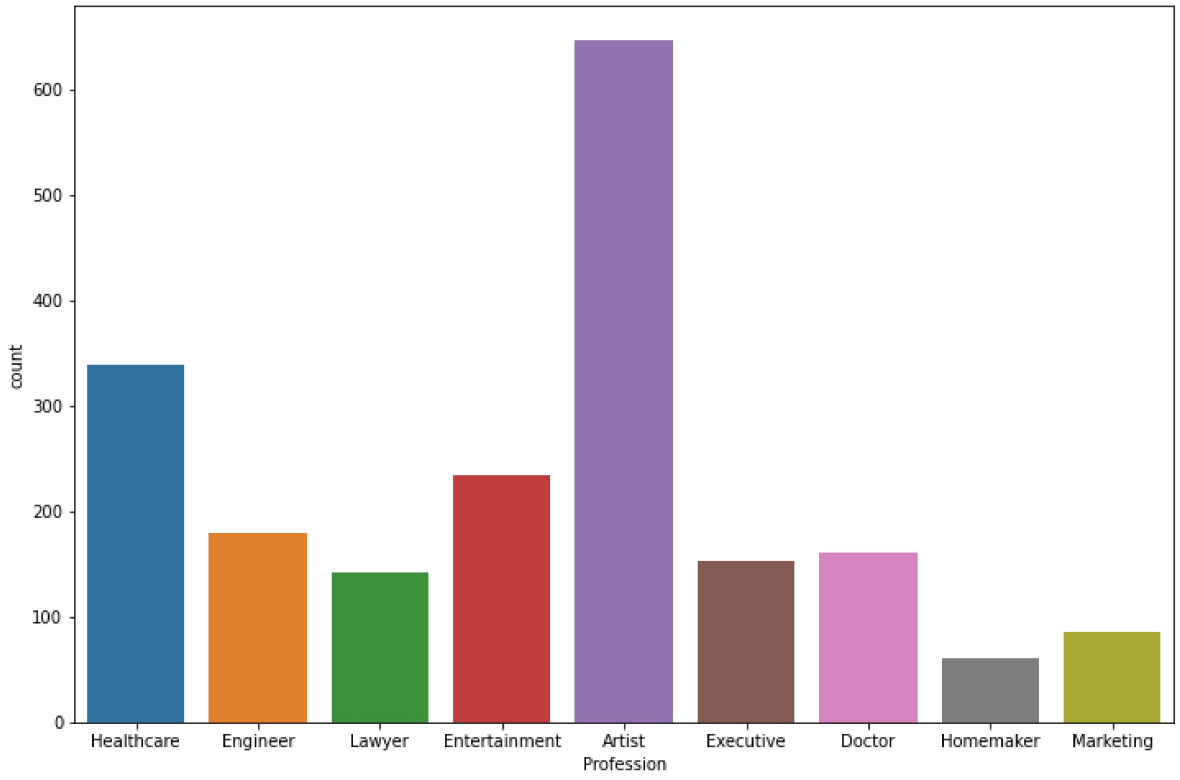
In [24]:
```
# categorical columns

cat_col= [feature for feature in df.columns if  df[feature].dtypes=="object"]
cat_col
```

Out[24]:
```
['Gender', 'Profession']
```

In [25]:
```
plt.figure(figsize=(12,6))
sns.countplot(df['Gender'])
plt.show()
```



In [26]:
```
plt.figure(figsize=(12,8))
sns.countplot(df['Profession'])
plt.show()
```



In [27]:
```
# getting the count of each categories from data

for feature in df.columns:
    print(df[feature].value_counts)
```

```
<bound method IndexOpsMixin.value_counts of 0          1
1          2
2          3
3          4
4          5
        ...
1995    1996
1996    1997
1997    1998
1998    1999
1999    2000
Name: CustomerID, Length: 2000, dtype: int64>
<bound method IndexOpsMixin.value_counts of 0         Male
1         Male
2       Female
3       Female
4       Female
         ...
1995    Female
1996    Female
1997      Male
1998      Male
1999      Male
Name: Gender, Length: 2000, dtype: object>
<bound method IndexOpsMixin.value_counts of 0       19
1       21
2       20
3       23
4       31
        ..
1995    71
1996    91
1997    87
1998    77
1999    90
Name: Age, Length: 2000, dtype: int64>
<bound method IndexOpsMixin.value_counts of 0        15000
1        35000
2        86000
3        59000
4        38000
         ...
1995    184387
1996     73158
1997     90961
1998    182109
1999    110610
Name: Annual Income ($), Length: 2000, dtype: int64>
<bound method IndexOpsMixin.value_counts of 0       39
1       81
2        6
3       77
4       40
        ..
1995    40
1996    32
1997    14
```

```
1998     4
1999    52
Name: Spending Score (1-100), Length: 2000, dtype: int64>
<bound method IndexOpsMixin.value_counts of 0          Healthcare
1            Engineer
2            Engineer
3              Lawyer
4       Entertainment
           ...
1995           Artist
1996           Doctor
1997       Healthcare
1998        Executive
1999    Entertainment
Name: Profession, Length: 2000, dtype: object>
<bound method IndexOpsMixin.value_counts of 0       1
1       3
2       1
3       0
4       2
       ..
1995    8
1996    7
1997    9
1998    7
1999    5
Name: Work Experience, Length: 2000, dtype: int64>
<bound method IndexOpsMixin.value_counts of 0       4
1       3
2       1
3       2
4       6
       ..
1995    7
1996    7
1997    2
1998    2
1999    2
Name: Family Size, Length: 2000, dtype: int64>
```

In [28]:
```
# Getting the counts of each categories from unique values

for feature in df.columns:
    print(df[feature].unique())
```

```
[   1    2    3 ... 1998 1999 2000]
['Male' 'Female']
[19 21 20 23 31 22 35 64 30 67 58 24 37 52 25 46 54 29 45 40 60 53 18 49
 42 36 65 48 50 27 33 59 47 51 69 70 63 43 68 32 26 57 38 55 34 66 39 44
 28 56 41 16 76 62 80  1  0 86 79 83 95 93 78 15  6 84  4 91 14 92 77 89
 12  7 94 96 74 85 73  9 10 11 17 90 61 13 72  5 75 99 88 82  8 87  3 97
 81 98  2 71]
[ 15000  35000  86000 ...  90961 182109 110610]
[ 39  81   6  77  40  76  94   3  72  14  99  15  13  79  35  66  29  98
  73   5  82  32  61  31  87   4  92  17  26  75  36  28  65  55  47  42
  52  60  54  45  41  50  46  51  56  59  48  49  53  44  57  58  43  91
  95  11   9  34  71  88   7  10  93  12  97  74  22  90  20  16  89   1
  78  83  27  63  86  69  24  68  85  23   8  18   0  33  70  37  64  30
  96   2  38  21  84  62  80 100  67  19  25]
['Healthcare' 'Engineer' 'Lawyer' 'Entertainment' 'Artist' 'Executive'
 'Doctor' 'Homemaker' 'Marketing']
[ 1  3  0  2  4  9 12 13  5  8 14  7  6 10 11 15 16 17]
[4 3 1 2 6 5 8 7 9]
```

Descriptive Statistics

In [29]:
```
df.describe()
```

Out[29]:

	CustomerID	Age	Annual Income ($)	Spending Score (1-100)	Work Experience	Family Size
count	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000	2000.000000
mean	1000.500000	48.960000	110731.821500	50.962500	4.102500	3.768500
std	577.494589	28.429747	45739.536688	27.934661	3.922204	1.970749
min	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000
25%	500.750000	25.000000	74572.000000	28.000000	1.000000	2.000000
50%	1000.500000	48.000000	110045.000000	50.000000	3.000000	4.000000
75%	1500.250000	73.000000	149092.750000	75.000000	7.000000	5.000000
max	2000.000000	99.000000	189974.000000	100.000000	17.000000	9.000000

In [30]:
```
# check the descriptive statistics of categorical_columns

df.describe(include='object')
```

Out[30]:

	Gender	Profession
count	2000	2000
unique	2	9
top	Female	Artist
freq	1186	647

In [31]:
```
# check the transpose statistics

df.describe().T
```

Out[31]:

	count	mean	std	min	25%	50%	75%	max
CustomerID	2000.0	1000.5000	577.494589	1.0	500.75	1000.5	1500.25	2000.0
Age	2000.0	48.9600	28.429747	0.0	25.00	48.0	73.00	99.0
Annual Income ($)	2000.0	110731.8215	45739.536688	0.0	74572.00	110045.0	149092.75	189974.0
Spending Score (1-100)	2000.0	50.9625	27.934661	0.0	28.00	50.0	75.00	100.0
Work Experience	2000.0	4.1025	3.922204	0.0	1.00	3.0	7.00	17.0
Family Size	2000.0	3.7685	1.970749	1.0	2.00	4.0	5.00	9.0

In [32]:
```
# check the correlation

df.corr()
```

Out[32]:

	CustomerID	Age	Annual Income ($)	Spending Score (1-100)	Work Experience	Family Size
CustomerID	1.000000	0.070700	0.328400	0.018936	0.091574	0.159655
Age	0.070700	1.000000	0.021378	-0.041798	-0.014319	0.038254
Annual Income ($)	0.328400	0.021378	1.000000	0.023299	0.089136	0.093005
Spending Score (1-100)	0.018936	-0.041798	0.023299	1.000000	-0.028948	0.002232
Work Experience	0.091574	-0.014319	0.089136	-0.028948	1.000000	0.011873
Family Size	0.159655	0.038254	0.093005	0.002232	0.011873	1.000000

In [33]:
```
# check the covariance

df.cov()
```

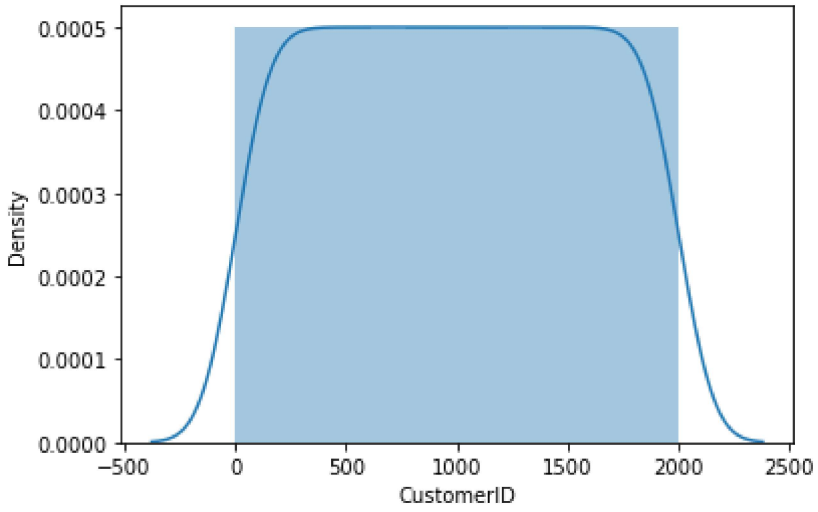
Out[33]:

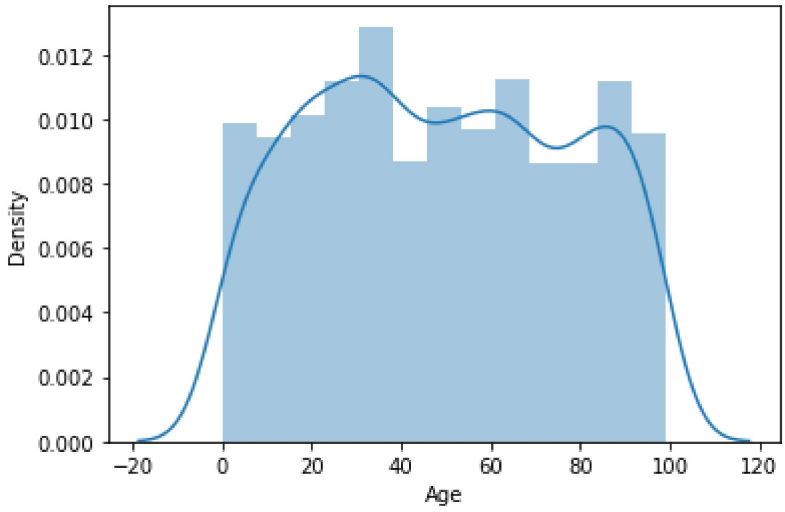
	CustomerID	Age	Annual Income ($)	Spending Score (1-100)	Work Experience	Family Size
CustomerID	3.335000e+05	1160.760380	8.674478e+06	305.469985	207.419460	181.703102
Age	1.160760e+03	808.250525	2.779861e+04	-33.195098	-1.596698	2.143312
Annual Income ($)	8.674478e+06	27798.605663	2.092105e+09	29770.078846	15991.046820	8383.595975
Spending Score (1-100)	3.054700e+02	-33.195098	2.977008e+04	780.345266	-3.171742	0.122880
Work Experience	2.074195e+02	-1.596698	1.599105e+04	-3.171742	15.383686	0.091775
Family Size	1.817031e+02	2.143312	8.383596e+03	0.122880	0.091775	3.883850

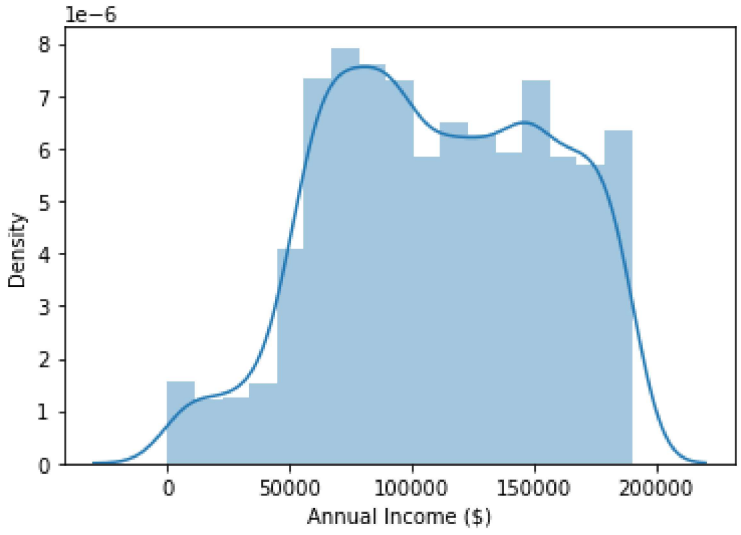
In [34]:
```
# check the skewness
df.skew()
```

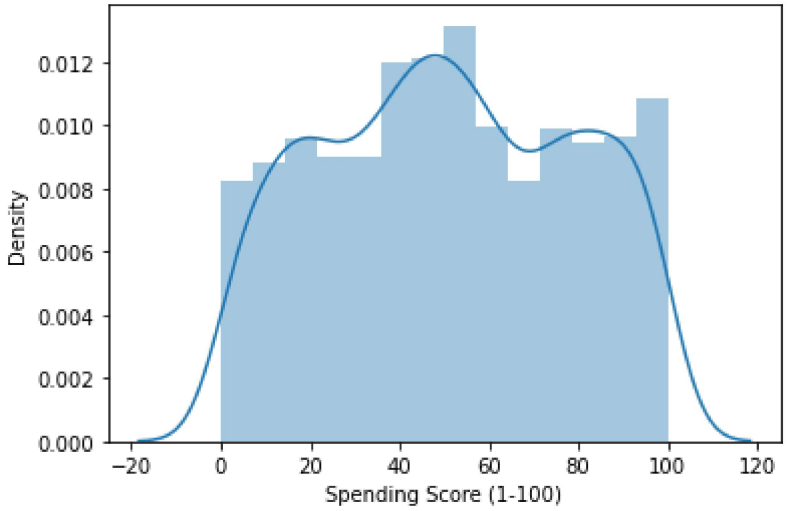
Out[34]:
```
CustomerID              0.000000
Age                     0.049222
Annual Income ($)      -0.116491
Spending Score (1-100)  0.004555
Work Experience         0.683718
Family Size             0.199263
dtype: float64
```

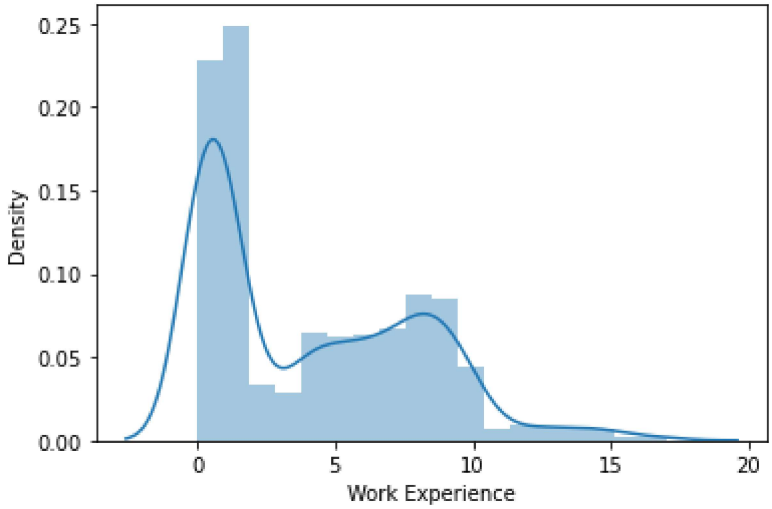
In [35]:
```
# check the distribution of numerical_columns

for i in num_col:
    sns.distplot(df[i])
    plt.show()
```

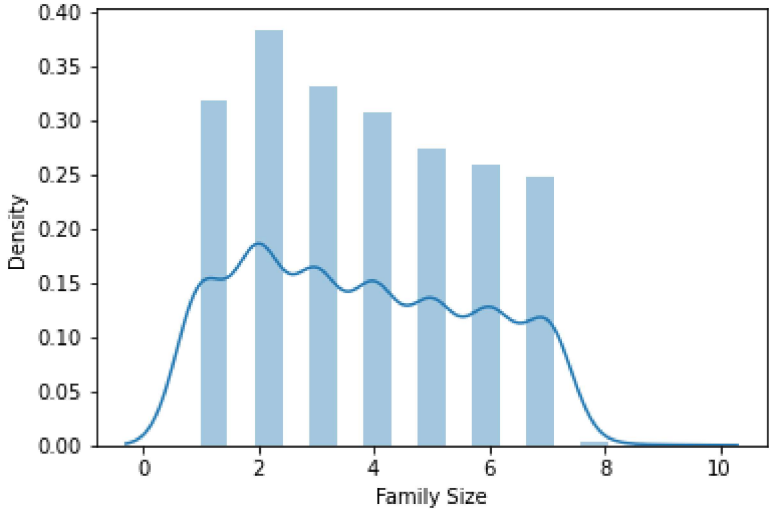







Density
0.012
0.010
0.008
0.006
0.004
0.002
0.000
−20
0
20
40
60
80
100
120
Spending Score (1-100)


Density
0.25
0.20
0.15
0.10
0.05
0.00
0
5
10
15
20
Work Experience


Density
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
2
4
6
8
10
Family Size

In [36]:
```
# check the quantile  values

df.quantile()
```

Out[36]:
```
CustomerID                  1000.5
Age                           48.0
Annual Income ($)         110045.0
Spending Score (1-100)        50.0
Work Experience                3.0
Family Size                    4.0
Name: 0.5, dtype: float64
```

In [37]:
```
# check the minimum values

df.min()
```

Out[37]:
```
CustomerID                     1
Gender                    Female
Age                            0
Annual Income ($)              0
Spending Score (1-100)         0
Profession                Artist
Work Experience                0
Family Size                    1
dtype: object
```

In [38]:
```
# check the maximum values

df.max()
```

Out[38]:
```
CustomerID                  2000
Gender                      Male
Age                           99
Annual Income ($)         189974
Spending Score (1-100)       100
Profession                Marketing
Work Experience                17
Family Size                     9
dtype: object
```

In [39]:
```
# check the mean() values

df.mean()
```

Out[39]:
```
CustomerID                  1000.5000
Age                           48.9600
Annual Income ($)         110731.8215
Spending Score (1-100)        50.9625
Work Experience                4.1025
Family Size                    3.7685
dtype: float64
```

In [40]:
```
# check the median() values

df.median()
```

Out[40]:
```
CustomerID                 1000.5
Age                          48.0
Annual Income ($)        110045.0
Spending Score (1-100)       50.0
Work Experience               3.0
Family Size                   4.0
dtype: float64
```

In [41]:
```
df.mode()
```

Out[41]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
0	1	Female	31.0	9000.0	49.0	Artist	1.0	2.0
1	2	NaN	NaN	50000.0	NaN	NaN	NaN	NaN
2	3	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	4	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	5	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...
1995	1996	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1996	1997	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1997	1998	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1998	1999	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1999	2000	NaN	NaN	NaN	NaN	NaN	NaN	NaN

2000 rows × 8 columns

In [42]:
```
df.columns
```

Out[42]:
```
Index(['CustomerID', 'Gender', 'Age', 'Annual Income ($)',
       'Spending Score (1-100)', 'Profession', 'Work Experience',
       'Family Size'],
      dtype='object')
```

In [43]:
```
df.head()
```

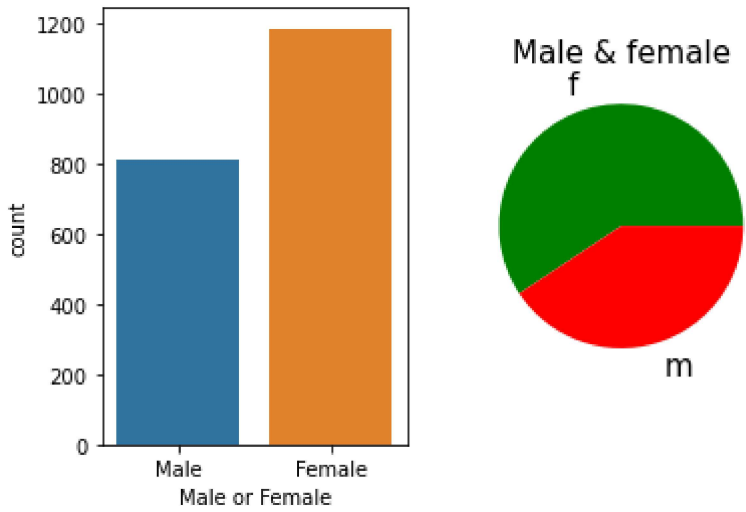
Out[43]:

	CustomerID	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
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

Target Columns

In [44]:
```
# gender checking

plt.subplot(1,2,1)
sns.countplot(df['Gender'])
plt.xlabel('Male or Female')

plt.subplot(1,2,2)
df.Gender.value_counts().plot(kind="pie",explode=[0,0],labels=["f","m"],fontsi
 , colors=["green","red"]                                )
plt.ylabel(None)
plt.title("Male & female",fontsize=15 )
```

Out[44]:
```
Text(0.5, 1.0, 'Male & female')
```



In [45]:
```
df['Gender'].value_counts()
```

Out[45]:
```
Female    1186
Male       814
Name: Gender, dtype: int64
```

In [46]:
```
df['Gender'].unique()
```

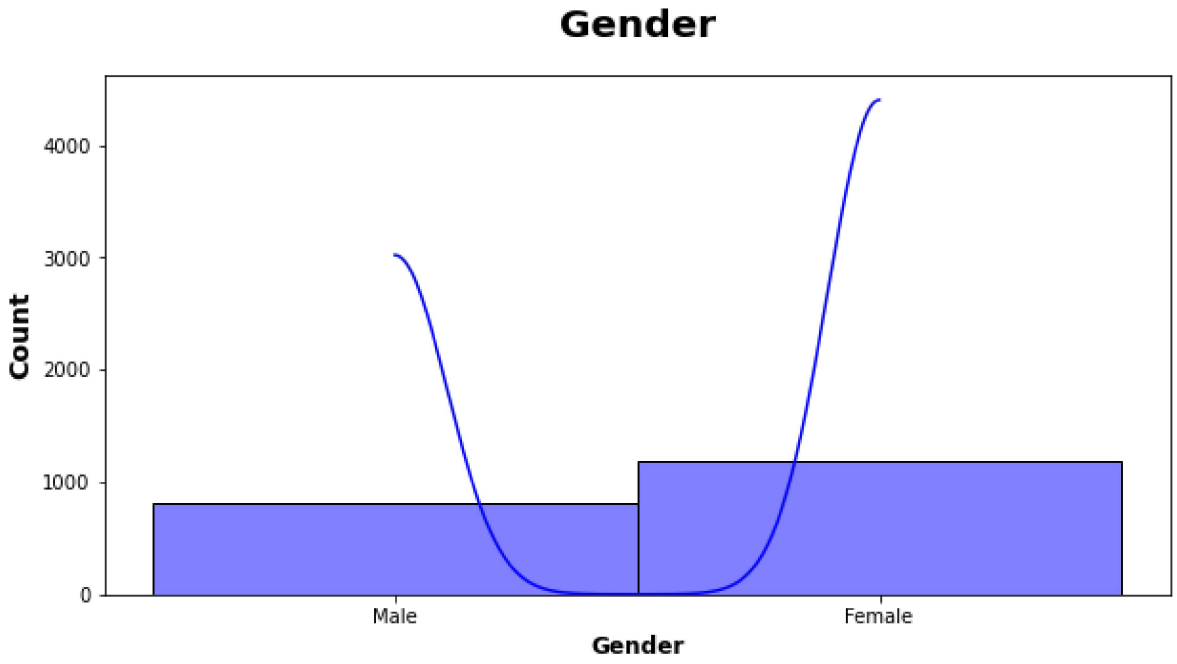
Out[46]:
```
array(['Male', 'Female'], dtype=object)
```

"Gender" data is imblanced data

We need a handle of imblance data

Visualized the Target Variable

In [47]:
```
plt.subplots(figsize=(10,5))
sns.histplot(df['Gender'],ec='Black',color='blue',kde=True)
plt.title('Gender', weight='bold',fontsize=20,pad=20)
plt.ylabel('Count',weight='bold',fontsize=14)
plt.xlabel('Gender',weight='bold',fontsize=12)
plt.show()
```



- Univariate Analysis
- Bivariate Analysis
- Multivariate Analysis

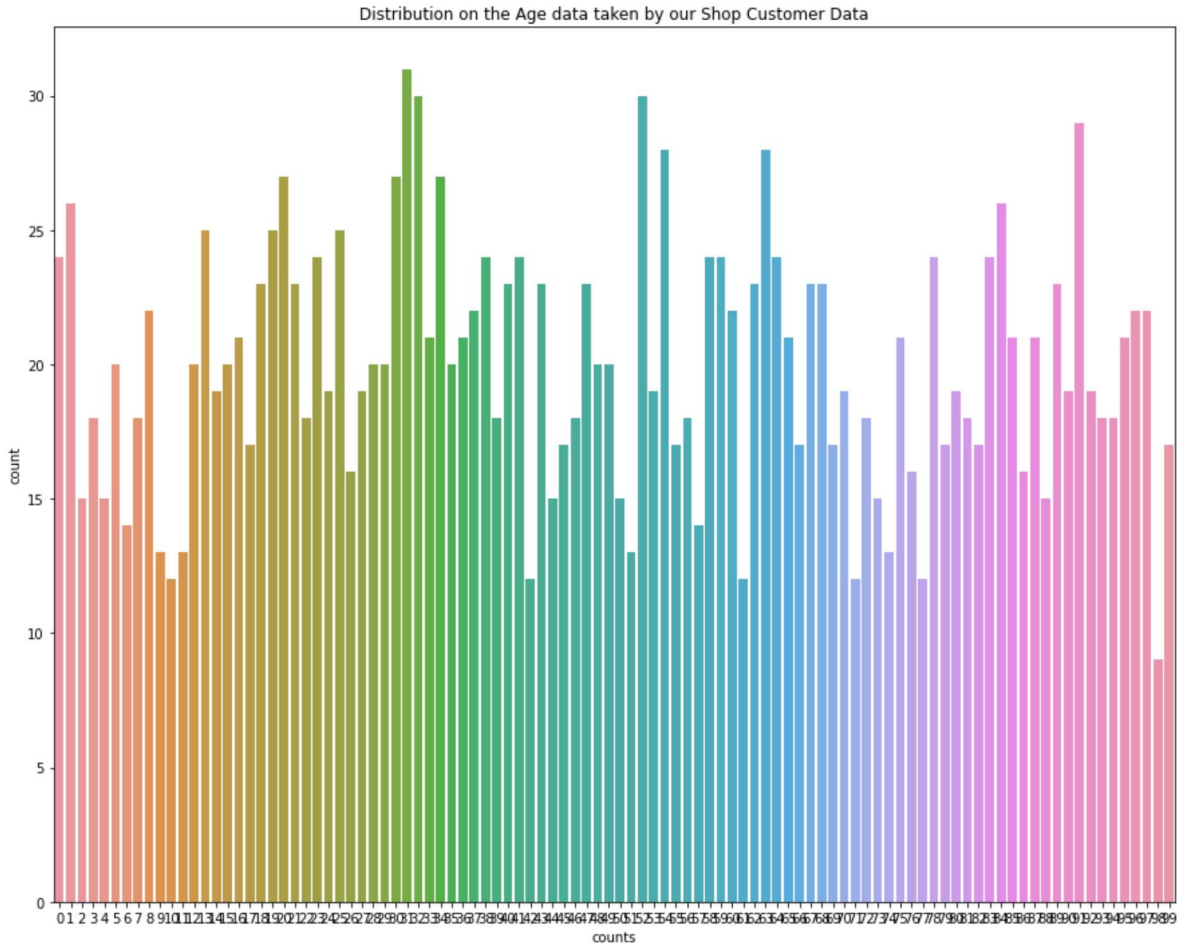
In [48]:
```
num_col
```

Out[48]:
```
['CustomerID',
 'Age',
 'Annual Income ($)',
 'Spending Score (1-100)',
 'Work Experience',
 'Family Size']
```

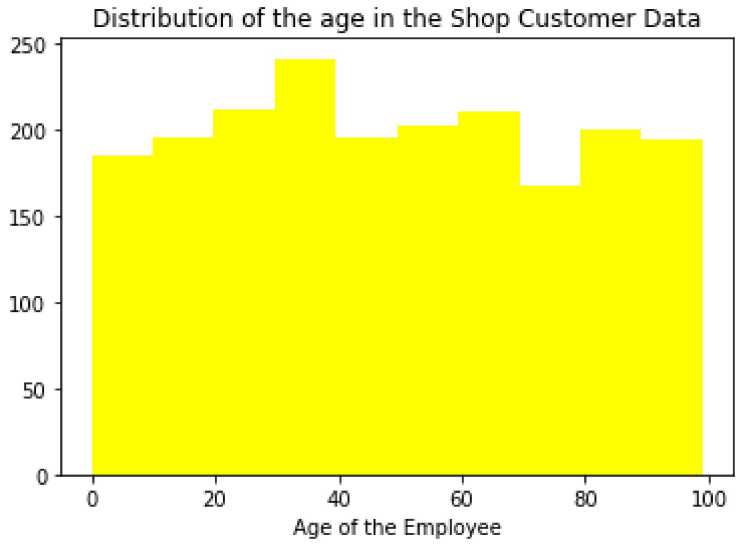
In [49]:
```
df['Age'].value_counts()
```

Out[49]:
```
31    31
32    30
52    30
91    29
63    28
      ..
42    12
10    12
77    12
71    12
98     9
Name: Age, Length: 100, dtype: int64
```

In [50]:
```
plt.figure(figsize=(15,12))
sns.countplot(df['Age'])
plt.xlabel('counts')
plt.title('Distribution on the Age data taken by our Shop Customer Data')
plt.show()
```

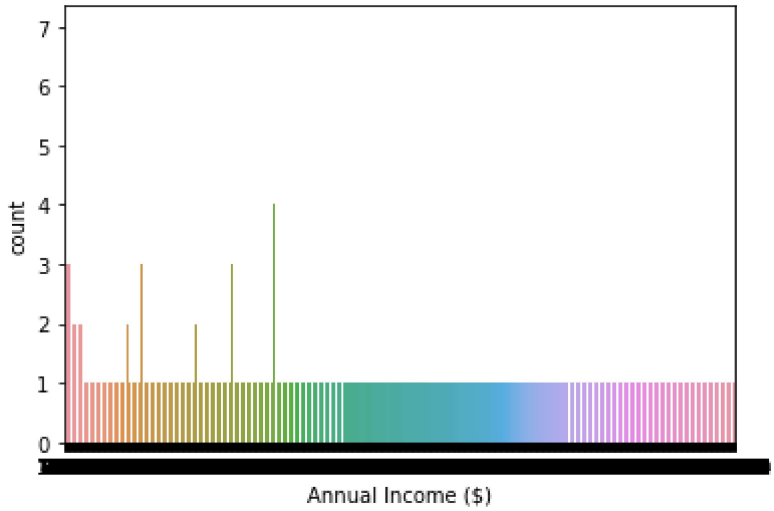


In [51]:
```
# histogram of age data
plt.hist(df['Age'],color='yellow')
plt.title('Distribution of the age in the Shop Customer Data')
plt.xlabel('Age of the Employee')
plt.show()
```



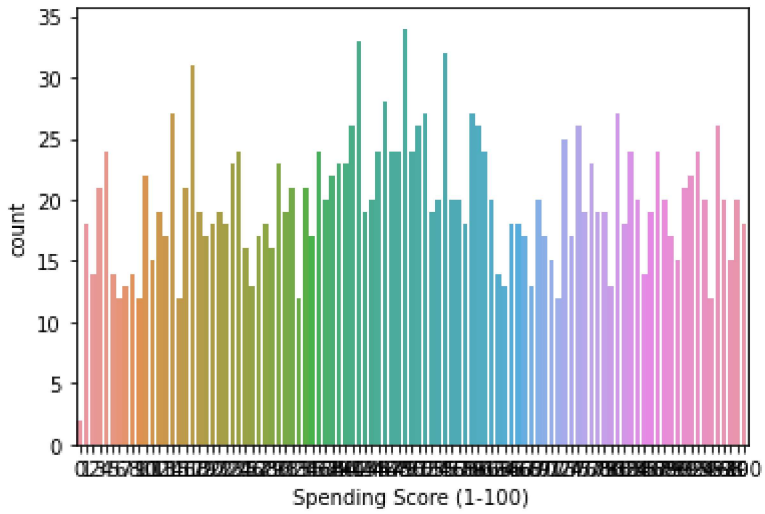
In [52]:
```
sns.countplot(df['Annual Income ($)'])
```

Out[52]:
```
<AxesSubplot:xlabel='Annual Income ($)', ylabel='count'>
```

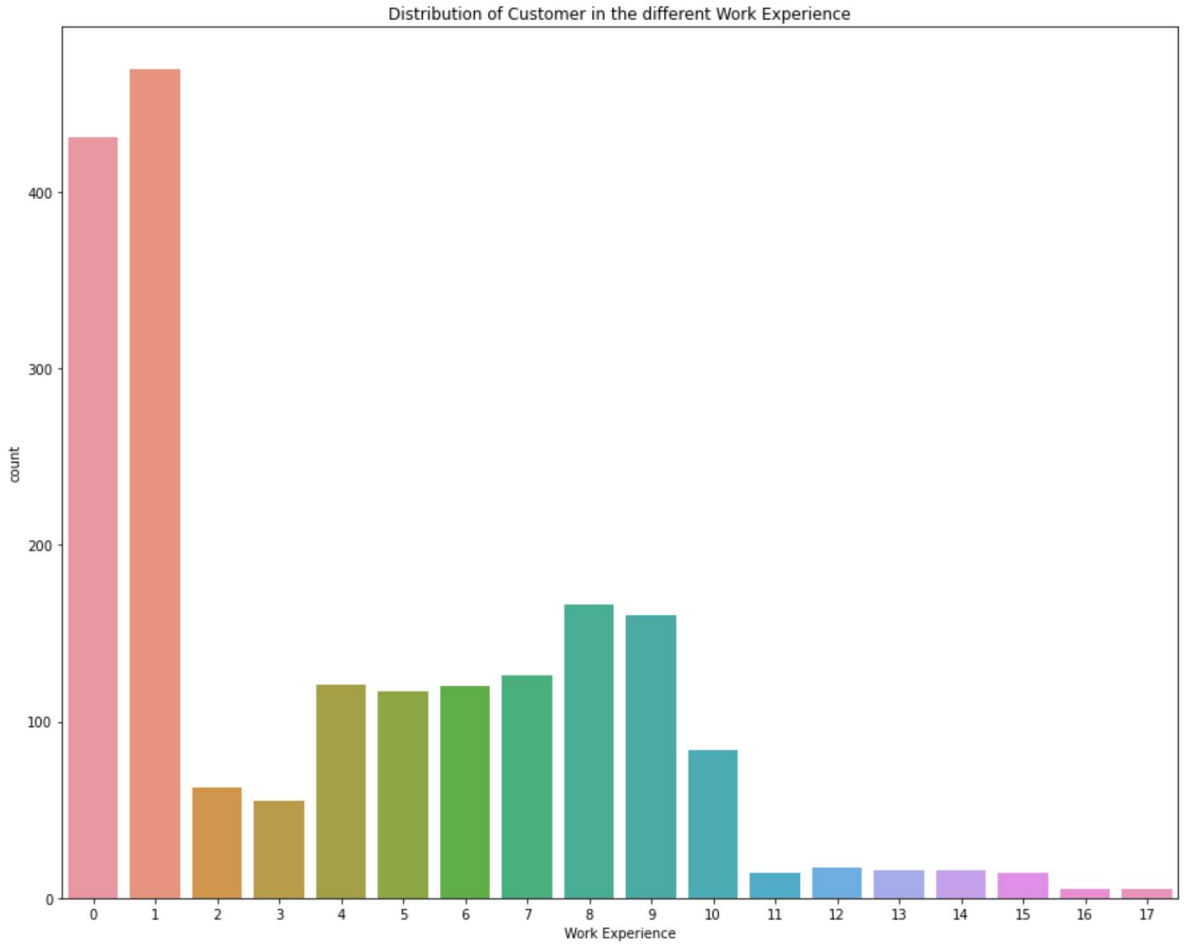


In [53]:
```
sns.countplot(df['Spending Score (1-100)'])
```

Out[53]: <AxesSubplot:xlabel='Spending Score (1-100)', ylabel='count'>

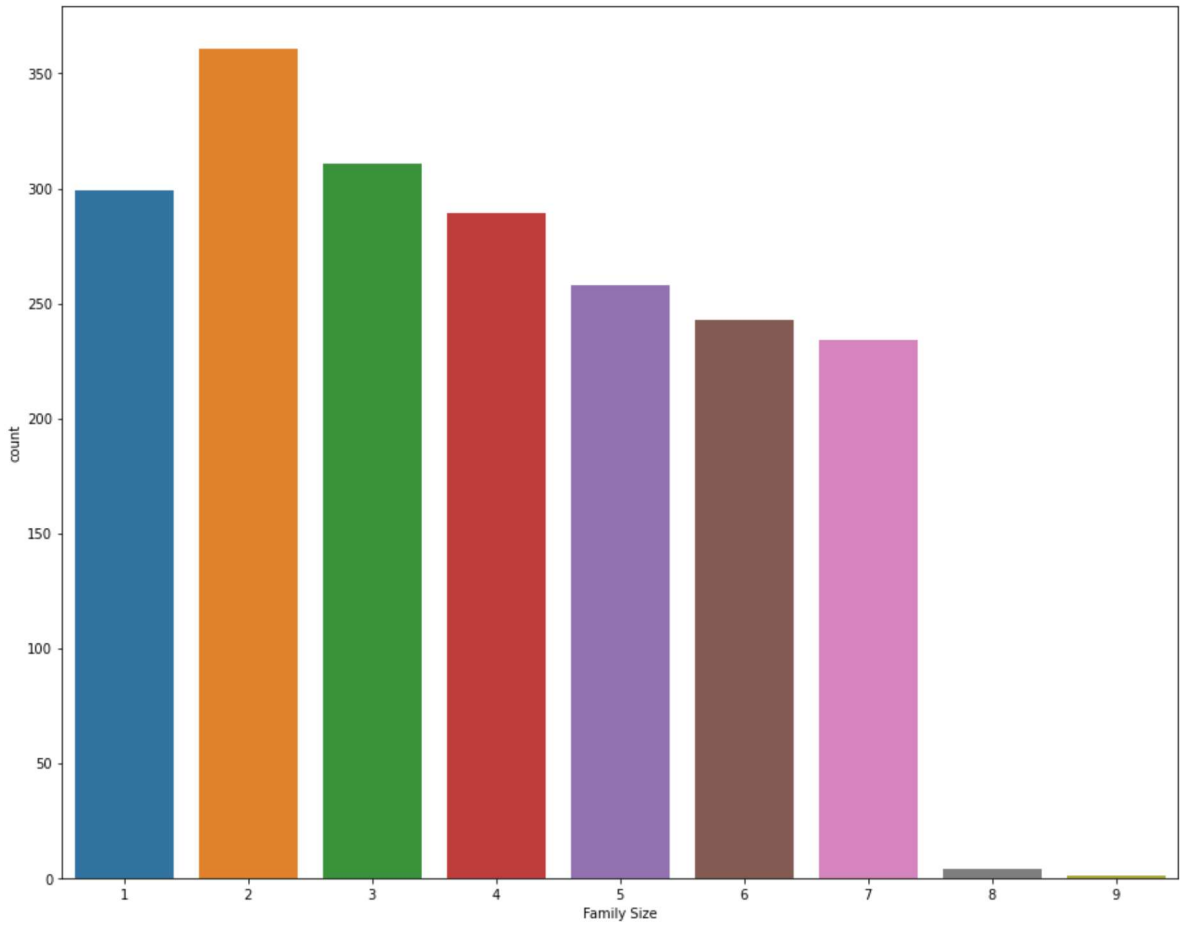


In [54]:
```
plt.figure(figsize=(15,12))
sns.countplot(df['Work Experience'])
plt.title('Distribution of Customer in the different Work Experience')
plt.show()
```



In [55]:
```
plt.figure(figsize=(15,12))
sns.countplot(df['Family Size'])
```

Out[55]: <AxesSubplot:xlabel='Family Size', ylabel='count'>

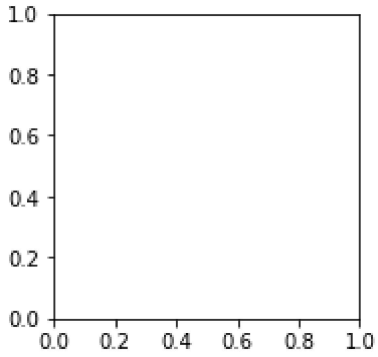


In [56]:
```
cat_col
```

Out[56]: ['Gender', 'Profession']

In [57]:
```
# Gender , Profession

plt.subplot(1,2,1)
labels=df['Gender'].value_counts().index
size = df['Gender'].value_counts
explode = None
plt.pie(size , labels=labels , explode=explode)
plt.title('Gender')

plt.subplot(1,2,2)
labels=df['Profession'].value_counts().index
size = df['Profession'].value_counts
explode = None
plt.pie(size , labels=labels , explode=explode)
plt.title('Gender')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [57], in <cell line: 7>()
      5 size = df['Gender'].value_counts
      6 explode = None
----> 7 plt.pie(size , labels=labels , explode=explode)
      8 plt.title('Gender')
     10 plt.subplot(1,2,2)

File ~\anaconda3\lib\site-packages\matplotlib\pyplot.py:2744, in pie(x, explo
de, labels, colors, autopct, pctdistance, shadow, labeldistance, startangle,
radius, counterclock, wedgeprops, textprops, center, frame, rotatelabels, nor
malize, data)
   2737 @_copy_docstring_and_deprecators(Axes.pie)
   2738 def pie(
   2739         x, explode=None, labels=None, colors=None, autopct=None,
   (...)
   2742         textprops=None, center=(0, 0), frame=False,
   2743         rotatelabels=False, *, normalize=True, data=None):
-> 2744     return gca().pie(
   2745         x, explode=explode, labels=labels, colors=colors,
   2746         autopct=autopct, pctdistance=pctdistance, shadow=shadow,
   2747         labeldistance=labeldistance, startangle=startangle,
   2748         radius=radius, counterclock=counterclock,
   2749         wedgeprops=wedgeprops, textprops=textprops, center=center,
   2750         frame=frame, rotatelabels=rotatelabels, normalize=normalize,
   2751         **({"data": data} if data is not None else {}))

File ~\anaconda3\lib\site-packages\matplotlib\__init__.py:1412, in _preproces
s_data.<locals>.inner(ax, data, *args, **kwargs)
   1409 @functools.wraps(func)
   1410 def inner(ax, *args, data=None, **kwargs):
   1411     if data is None:
-> 1412         return func(ax, *map(sanitize_sequence, args), **kwargs)
   1414     bound = new_sig.bind(ax, *args, **kwargs)
   1415     auto_label = (bound.arguments.get(label_namer)
   1416                   or bound.kwargs.get(label_namer))

File ~\anaconda3\lib\site-packages\matplotlib\axes\_axes.py:3042, in Axes.pie
(self, x, explode, labels, colors, autopct, pctdistance, shadow, labeldistanc
e, startangle, radius, counterclock, wedgeprops, textprops, center, frame, ro
tatelabels, normalize)
   3039 self.set_aspect('equal')
   3040 # The use of float32 is "historical", but can't be changed without
   3041 # regenerating the test baselines.
-> 3042 x = np.asarray(x, np.float32)
   3043 if x.ndim > 1:
   3044     raise ValueError("x must be 1D")

TypeError: float() argument must be a string or a number, not 'method'
```



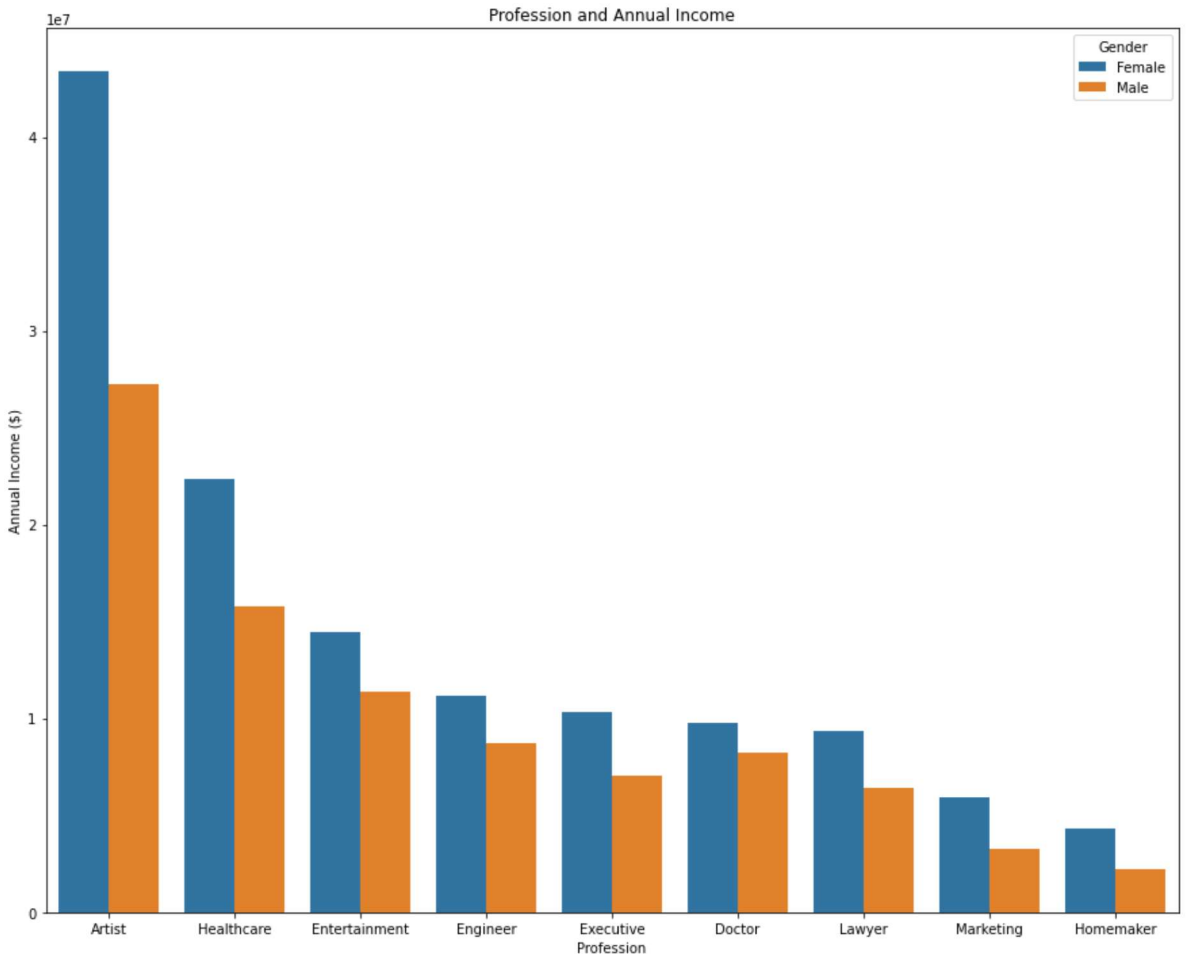
In [58]:
```
# Gender , Profession

df_profession = df.groupby(['Profession','Gender'])['Annual Income ($)'] .sum(
df_profession
```

Out[58]:

	Profession	Gender	Annual Income ($)
0	Artist	Female	43428525
1	Artist	Male	27245926
2	Healthcare	Female	22386044
3	Healthcare	Male	15776556
4	Entertainment	Female	14484697
5	Entertainment	Male	11407481
6	Engineer	Female	11175263
7	Executive	Female	10335668
8	Doctor	Female	9742222
9	Lawyer	Female	9374329
10	Engineer	Male	8722599
11	Doctor	Male	8221066
12	Executive	Male	7071162
13	Lawyer	Male	6387080
14	Marketing	Female	5896076
15	Homemaker	Female	4293882
16	Marketing	Male	3283432
17	Homemaker	Male	2231635

In [59]:
```
plt.figure(figsize=(15,12))
sns.barplot(data=df_profession, x= "Profession", y= "Annual Income ($)", hue="
plt.title('Profession and Annual Income')
```

Out[59]:
```
Text(0.5, 1.0, 'Profession and Annual Income')
```



In [60]:
```
# Age and Profession

df_age = df.groupby(['Age' , 'Profession'])['Annual Income ($)'] .sum().sort_v
df_age
```

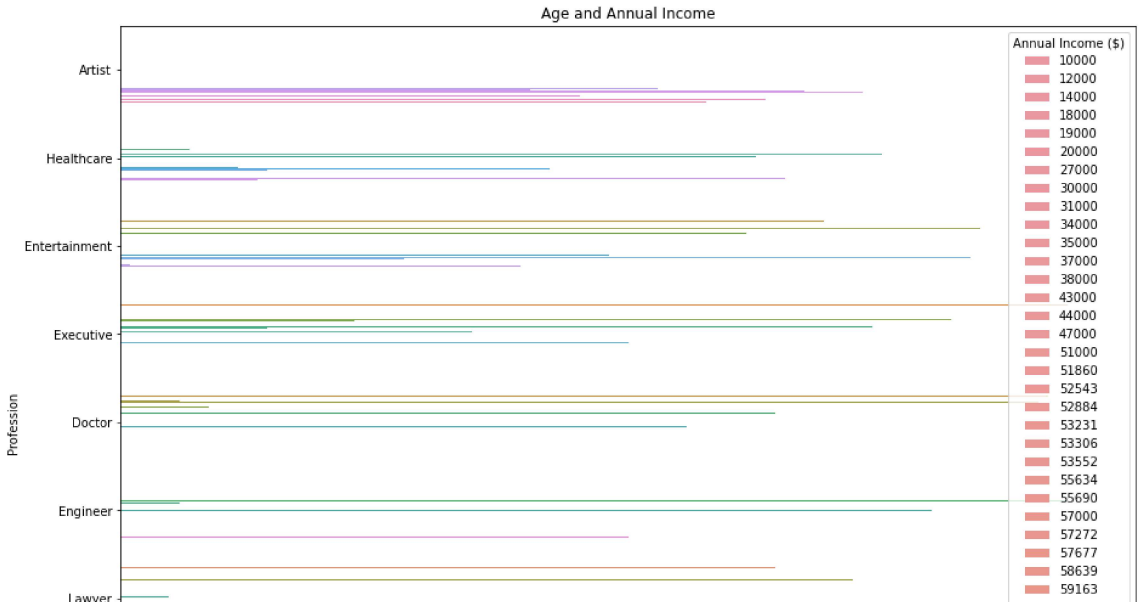
Out[60]:

	Age	Profession	Annual Income ($)
0	84	Artist	1444509
1	1	Artist	1412913
2	30	Artist	1349234
3	25	Artist	1303389
4	62	Artist	1276432
...	...	...	...
695	37	Doctor	19000
696	50	Marketing	18000
697	42	Doctor	14000
698	86	Marketing	12000
699	27	Marketing	10000

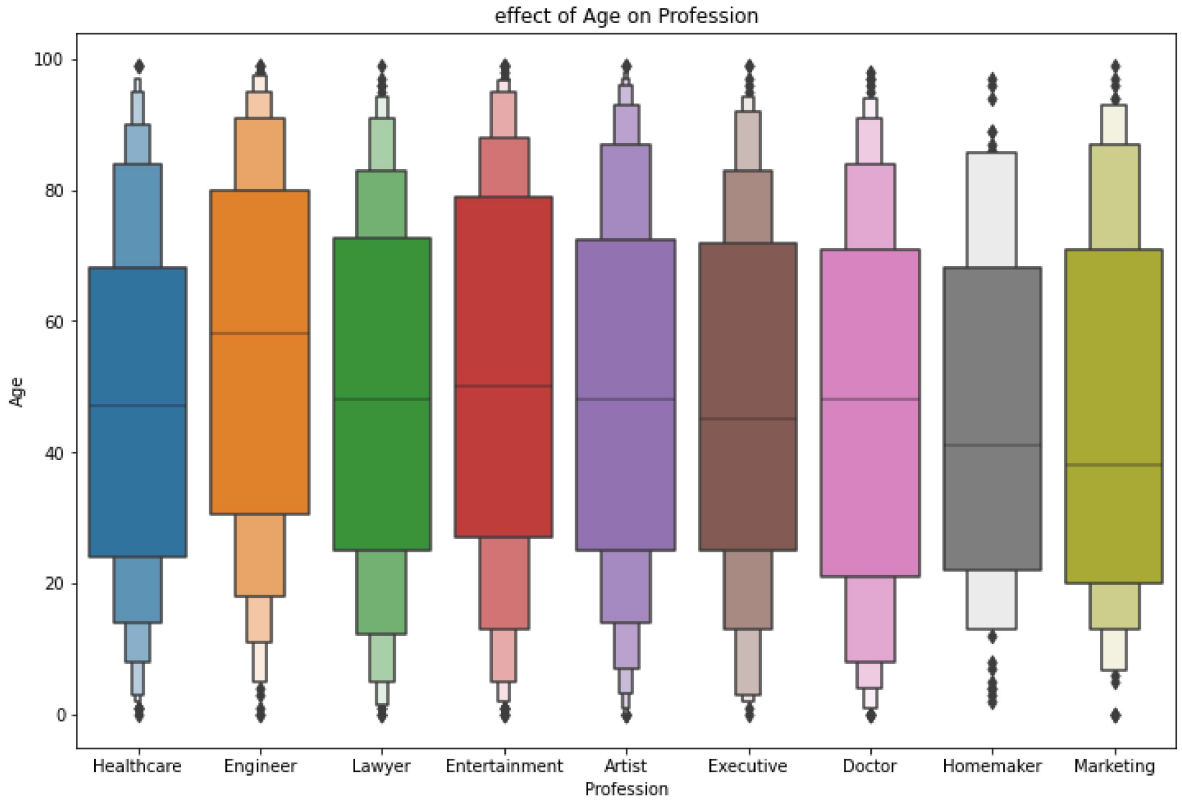
700 rows × 3 columns

In [61]:
```
plt.figure(figsize=(15,12))
sns.barplot(data=df_age ,x='Age' , y= 'Profession' , hue='Annual Income ($)')
plt.title("Age and Annual Income")
```

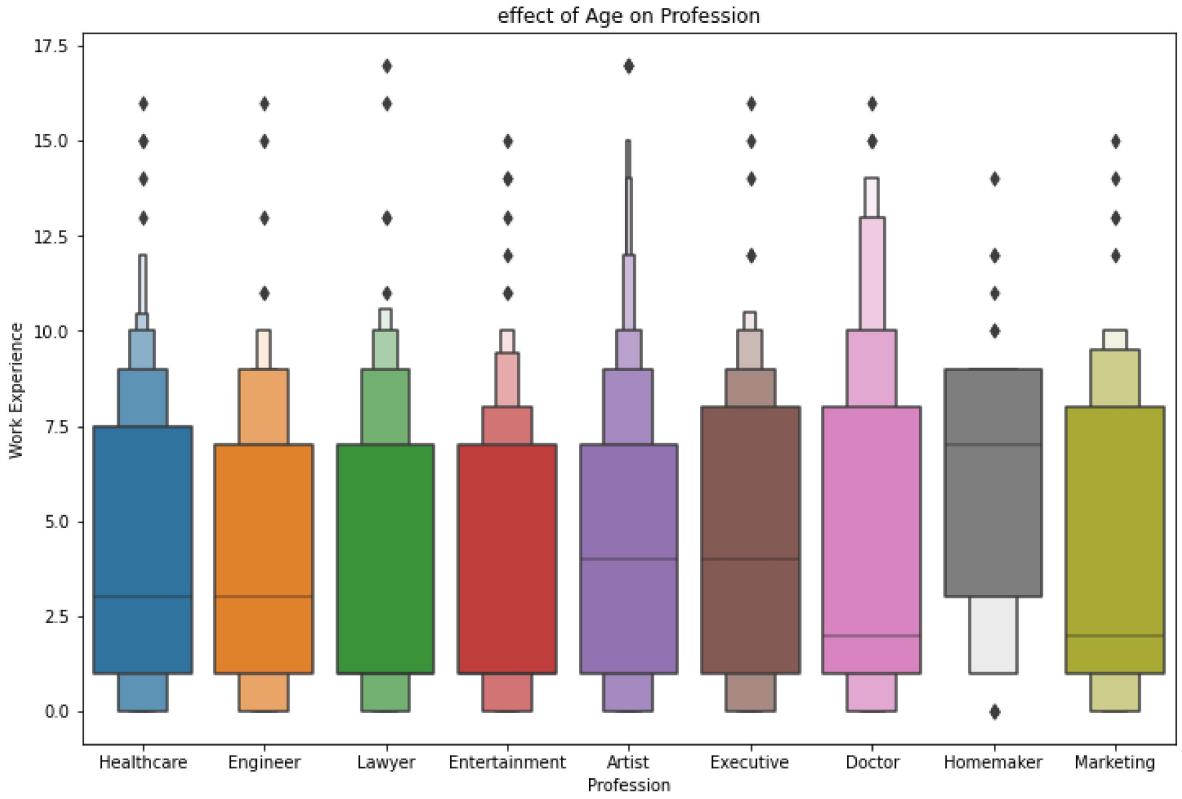
Out[61]:
```
Text(0.5, 1.0, 'Age and Annual Income')
```



In [62]:
```
# Age and Profession
plt.figure(figsize=(12,8))
sns.boxenplot(df['Profession'],df['Age'])
plt.title('effect of Age on Profession')
plt.show()
```



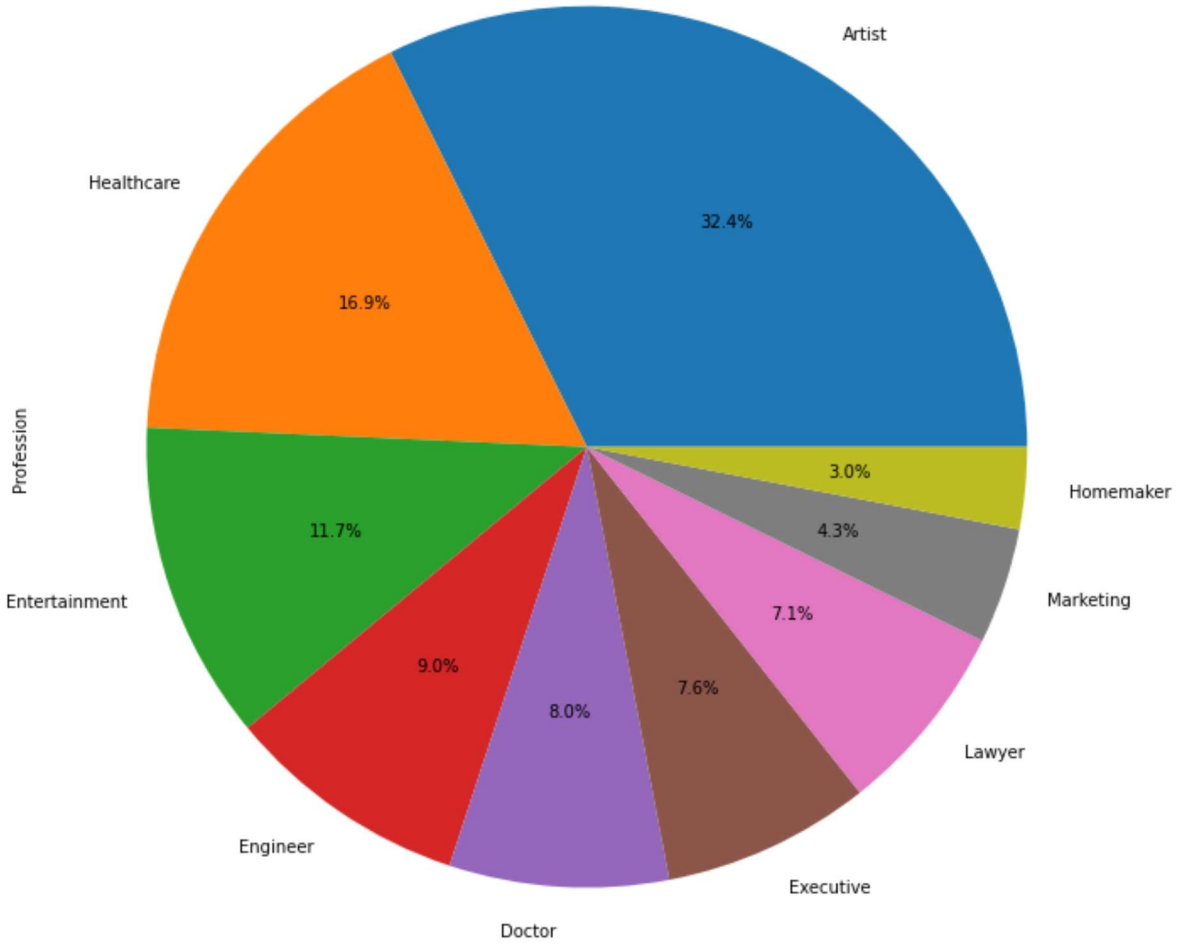
In [63]:
```
#  Profession and Work Experiance
plt.figure(figsize=(12,8))
sns.boxenplot(df['Profession'],df['Work Experience'])
plt.title('effect of Age on Profession')
plt.show()
```



Which one of the most popular Profession

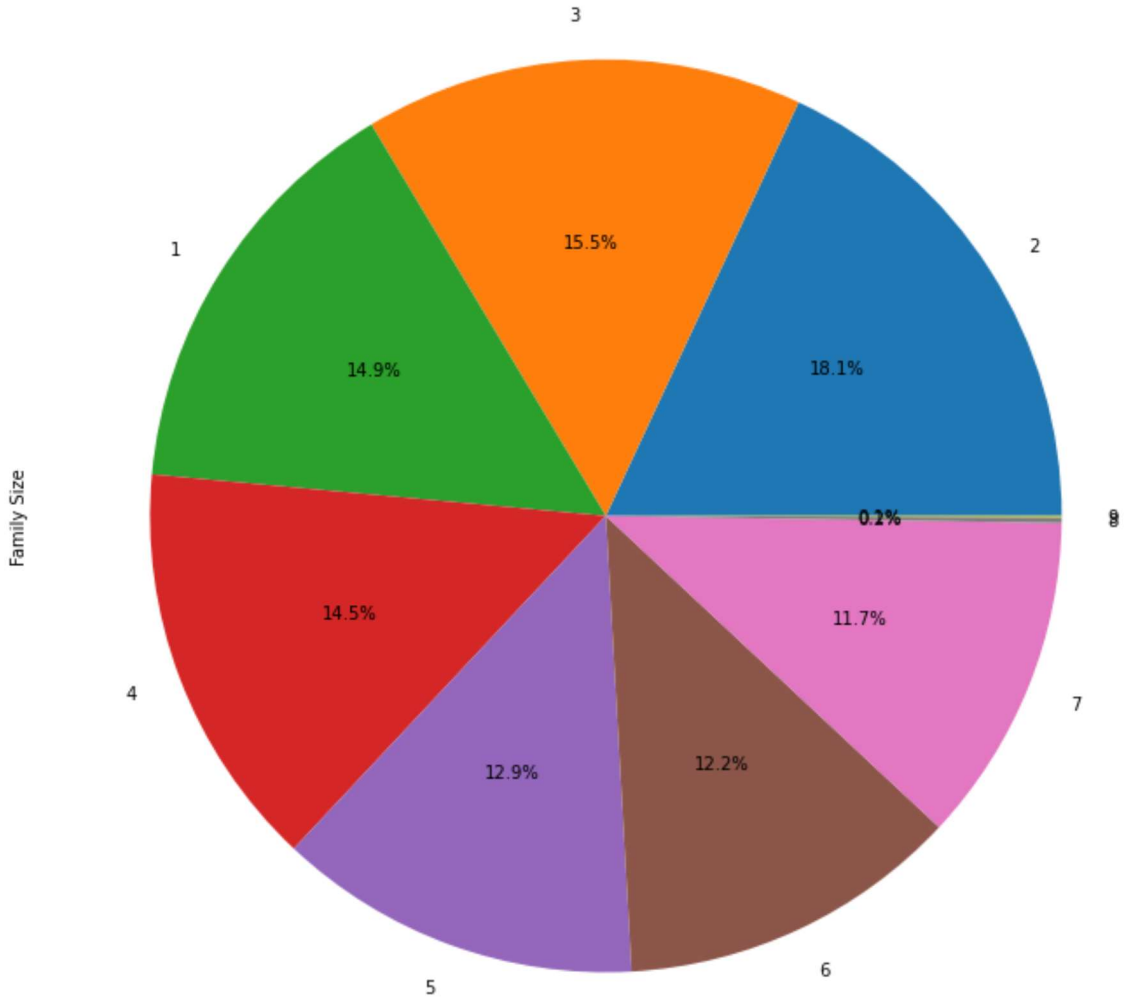
In [64]:
```
df['Profession'].value_counts().plot.pie(figsize=(15,12),autopct="%1.1f%%")
```

Out[64]: <AxesSubplot:ylabel='Profession'>



In [65]:
```
df['Family Size'].value_counts().plot.pie(figsize=(15,12),autopct="%1.1f%%")
```

Out[65]: <AxesSubplot:ylabel='Family Size'>



In [66]:
```
# Heat Map

df.corr()
```

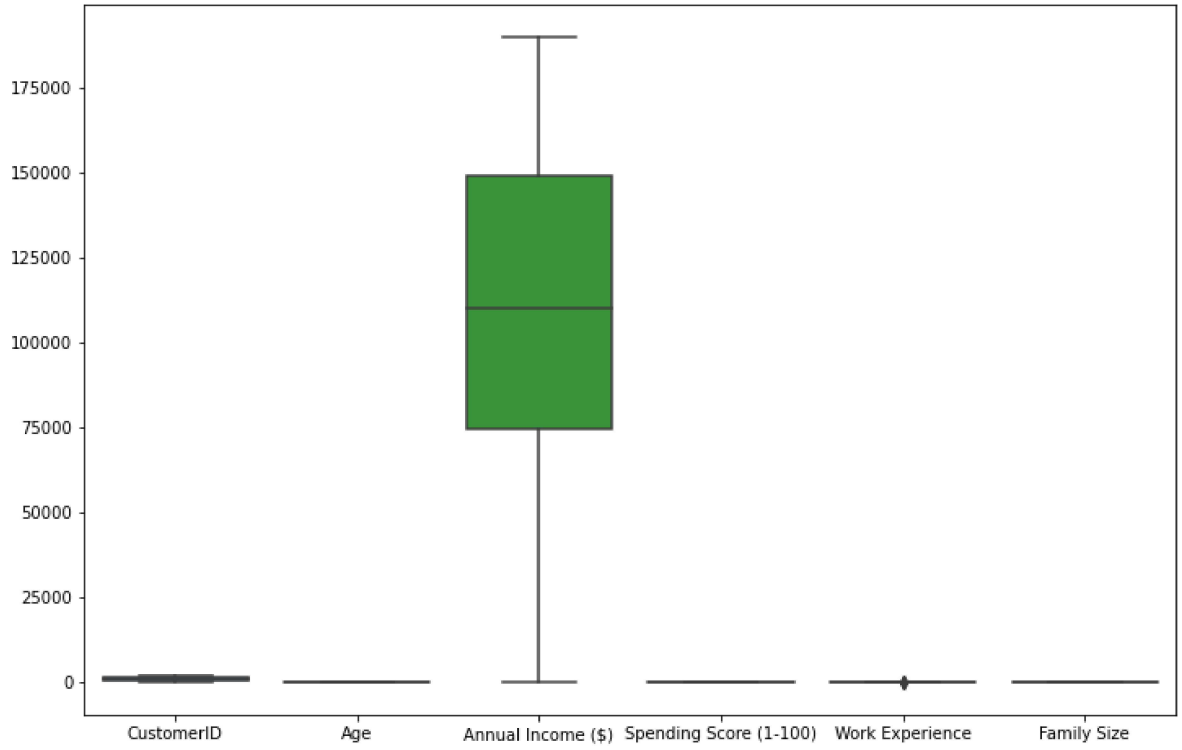
Out[66]:

	CustomerID	Age	Annual Income ($)	Spending Score (1-100)	Work Experience	Family Size
CustomerID	1.000000	0.070700	0.328400	0.018936	0.091574	0.159655
Age	0.070700	1.000000	0.021378	-0.041798	-0.014319	0.038254
Annual Income ($)	0.328400	0.021378	1.000000	0.023299	0.089136	0.093005
Spending Score (1-100)	0.018936	-0.041798	0.023299	1.000000	-0.028948	0.002232
Work Experience	0.091574	-0.014319	0.089136	-0.028948	1.000000	0.011873
Family Size	0.159655	0.038254	0.093005	0.002232	0.011873	1.000000

In [67]:
```
plt.figure(figsize=(12,8))
sns.heatmap(df.corr() , annot = True)
plt.show()
```

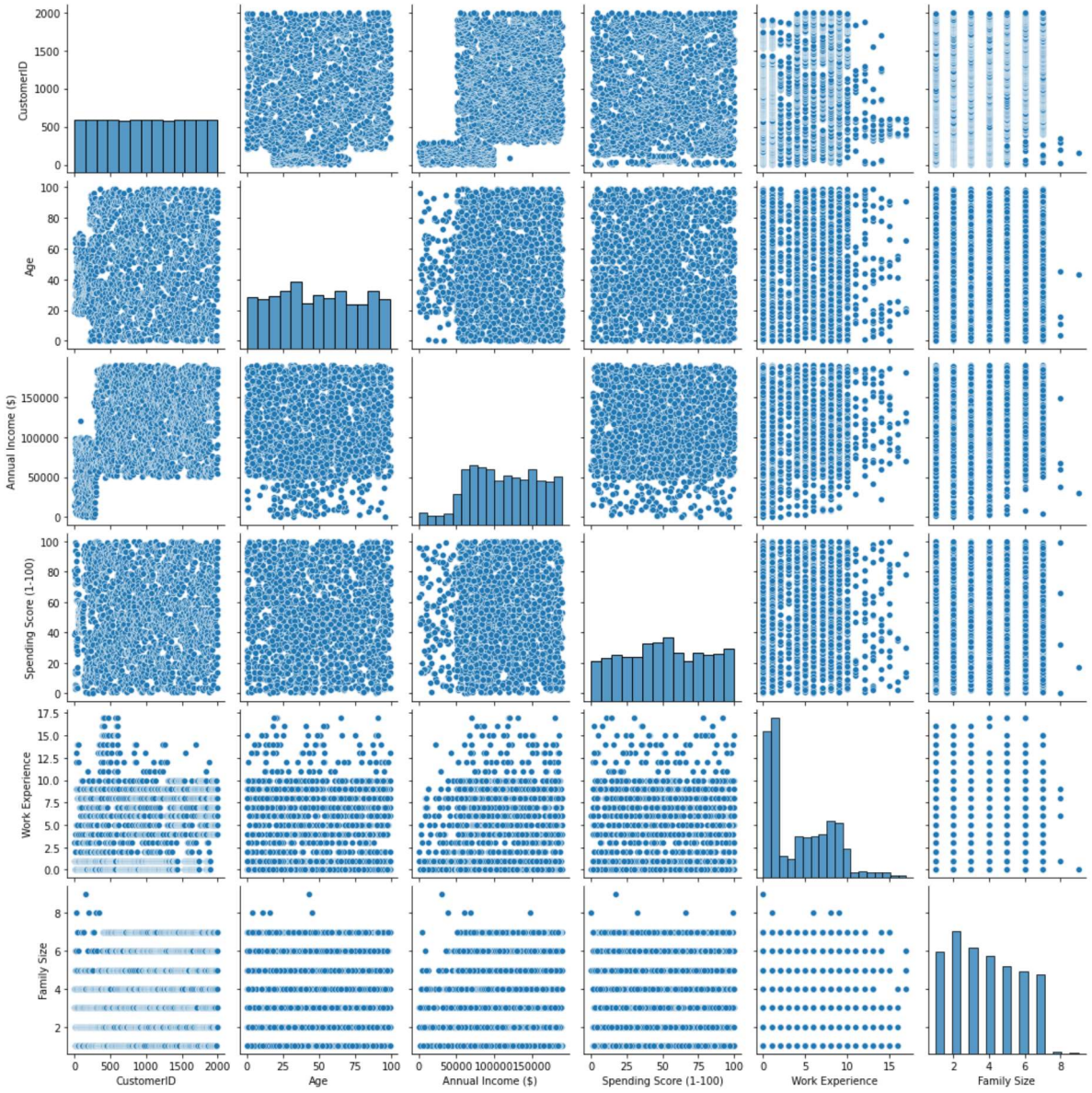
In [68]:
```
 # Box plot
plt.figure(figsize=(12,8))
sns.boxplot(data=df , orient = 'V')
```

Out[68]: <AxesSubplot:>

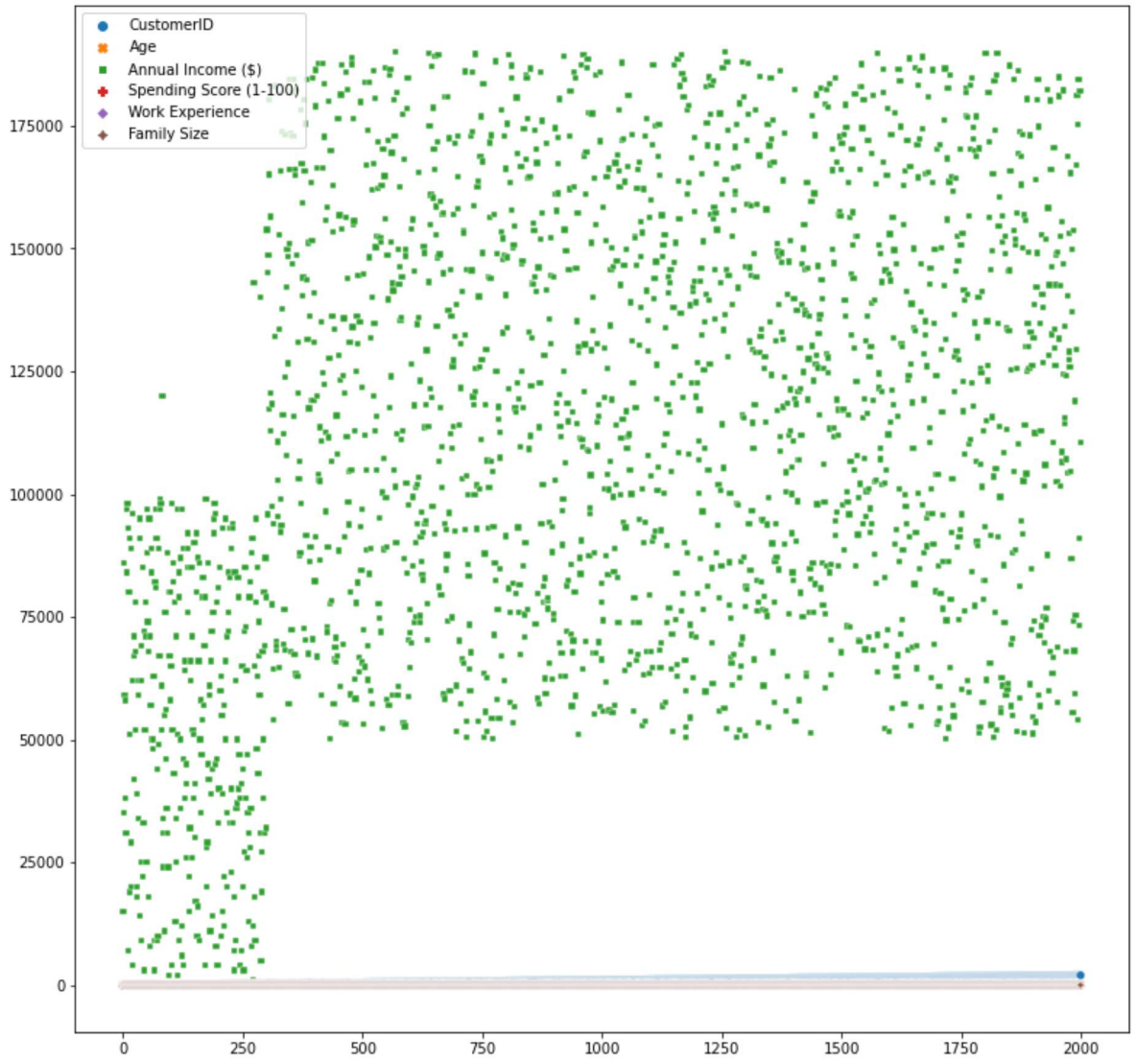


In [69]:
```
# pairplot

sns.pairplot(df)
```

Out[69]:
```
<seaborn.axisgrid.PairGrid at 0x1ddada407c0>
```



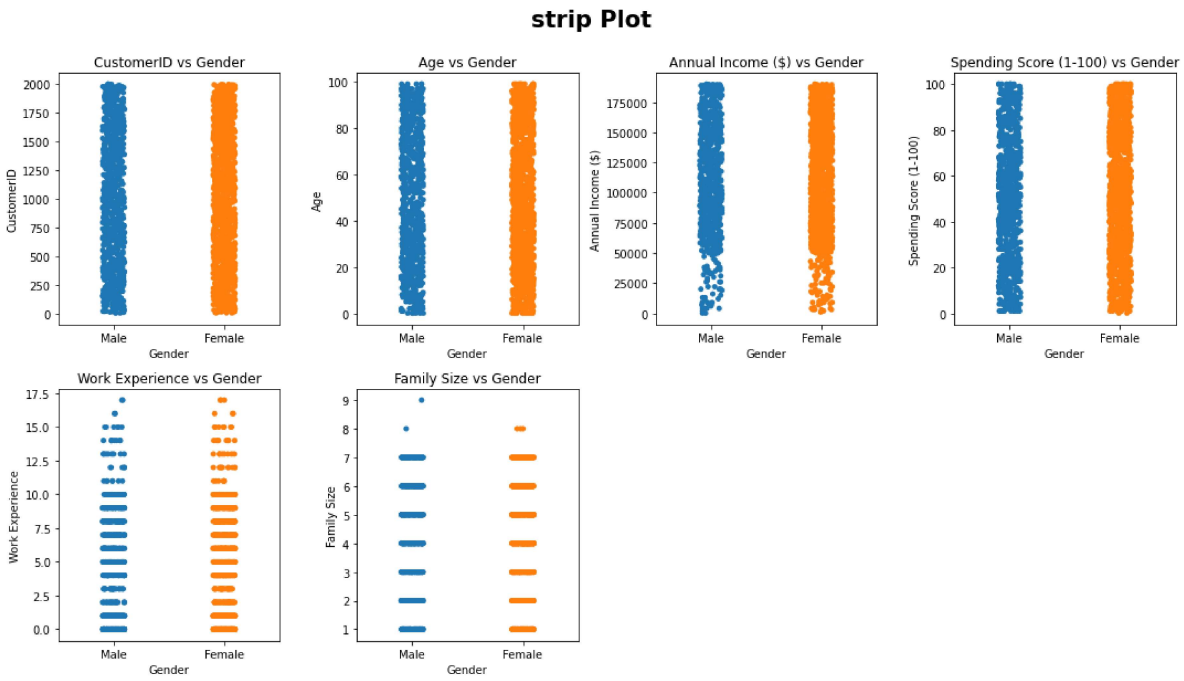
In [70]:
```
# scatter Plot

plt.figure(figsize=(13,13))
sns.scatterplot(data=df)
plt.show()
```



In [ ]:
```

```

Strip plot

In [71]:
```
#Visualising data scatter in each continous feature with respect to quality
plt.figure(figsize=(15,20))
plt.suptitle('strip Plot', fontsize =21 , fontweight ='bold' , alpha = 1 , y=
for i in range (0 , len(num_col)):
    plt.subplot(5,4,i+1)
    sns.stripplot(y= num_col[i], x='Gender' , data=df)
    plt.title('{} vs Gender'.format(num_col[i],fontsize=15))
    plt.tight_layout()
```



drop columns

In [72]:
```
df = df.drop(['CustomerID'],axis=1)
```

In [73]:
```
df.head()
```

Out[73]:

	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
0	Male	19	15000	39	Healthcare	1	4
1	Male	21	35000	81	Engineer	3	3
2	Female	20	86000	6	Engineer	1	1
3	Female	23	59000	77	Lawyer	0	2
4	Female	31	38000	40	Entertainment	2	6

In [74]:
```
# Checking the categorical columns to convert into numerical columns

df.select_dtypes('object')
```

Out[74]:

	Gender	Profession
0	Male	Healthcare
1	Male	Engineer
2	Female	Engineer
3	Female	Lawyer
4	Female	Entertainment
...	...	...
1995	Female	Artist
1996	Female	Doctor
1997	Male	Healthcare
1998	Male	Executive
1999	Male	Entertainment

2000 rows × 2 columns

Binary Encoding

In [75]:
```
df['Gender'] = df['Gender'].replace({'Male':0 , 'Female':1})
```

Label Encoder

In [76]:
```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [77]:
```
df['Profession'] = le.fit_transform(df['Profession'])
```

In [78]:
```
df.head()
```

Out[78]:

	Gender	Age	Annual Income ($)	Spending Score (1-100)	Profession	Work Experience	Family Size
0	0	19	15000	39	5	1	4
1	0	21	35000	81	2	3	3
2	1	20	86000	6	2	1	1
3	1	23	59000	77	7	0	2
4	1	31	38000	40	3	2	6

In [ ]: