2440016804 - Rio Pramana - LA01 - Assignment 5

Import libraries and load dataset

In [1]:
```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
```

In [2]:
```
# Importing the dataset, downloaded file is in the same folder
csv_path = "Social_Network_Ads.csv"
SNA_df = pd.read_csv(csv_path)
```

Check dataset (Shape, Info)

In [3]:
```
SNA_df.shape
```

Out[3]:
```
(400, 3)
```

In [4]:
```
SNA_df.head(10)
```

Out[4]:

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0
5	27	58000	0
6	27	84000	0

	Age	EstimatedSalary	Purchased
7	32	150000	1
8	25	33000	0
9	35	65000	0

In [5]:
```
SNA_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Age              400 non-null    int64
 1   EstimatedSalary  400 non-null    int64
 2   Purchased        400 non-null    int64
dtypes: int64(3)
memory usage: 9.5 KB
```

No missing data and anomaly of the data type found

Data Summarization

In [6]:
```
SNA_df.describe()
```

Out[6]:

	Age	EstimatedSalary	Purchased
count	400.000000	400.000000	400.000000
mean	37.655000	69742.500000	0.357500
std	10.482877	34096.960282	0.479864
min	18.000000	15000.000000	0.000000
25%	29.750000	43000.000000	0.000000
50%	37.000000	70000.000000	0.000000
75%	46.000000	88000.000000	1.000000

	Age	EstimatedSalary	Purchased
max	60.000000	150000.000000	1.000000

Extracting independent and dependent variables

In [7]:
```
#Extracting independent variables:
x = SNA_df.iloc[:, :-1].values #Extract semua kolom kecuali kolom terakhir
print(x)
```

```
[[    19  19000]
 [    35  20000]
 [    26  43000]
 [    27  57000]
 [    19  76000]
 [    27  58000]
 [    27  84000]
 [    32 150000]
 [    25  33000]
 [    35  65000]
 [    26  80000]
 [    26  52000]
 [    20  86000]
 [    32  18000]
 [    18  82000]
 [    29  80000]
 [    47  25000]
 [    45  26000]
 [    46  28000]
 [    48  29000]
 [    45  22000]
 [    47  49000]
 [    48  41000]
 [    45  22000]
 [    46  23000]
 [    47  20000]
 [    49  28000]
 [    47  30000]
 [    29  43000]
 [    31  18000]
 [    31  74000]
 [    27 137000]
```

```
[    21  16000]
[    28  44000]
[    27  90000]
[    35  27000]
[    33  28000]
[    30  49000]
[    26  72000]
[    27  31000]
[    27  17000]
[    33  51000]
[    35 108000]
[    30  15000]
[    28  84000]
[    23  20000]
[    25  79000]
[    27  54000]
[    30 135000]
[    31  89000]
[    24  32000]
[    18  44000]
[    29  83000]
[    35  23000]
[    27  58000]
[    24  55000]
[    23  48000]
[    28  79000]
[    22  18000]
[    32 117000]
[    27  20000]
[    25  87000]
[    23  66000]
[    32 120000]
[    59  83000]
[    24  58000]
[    24  19000]
[    23  82000]
[    22  63000]
[    31  68000]
[    25  80000]
[    24  27000]
[    20  23000]
[    33 113000]
[    32  18000]
[    34 112000]
```

```
[    18  52000]
[    22  27000]
[    28  87000]
[    26  17000]
[    30  80000]
[    39  42000]
[    20  49000]
[    35  88000]
[    30  62000]
[    31 118000]
[    24  55000]
[    28  85000]
[    26  81000]
[    35  50000]
[    22  81000]
[    30 116000]
[    26  15000]
[    29  28000]
[    29  83000]
[    35  44000]
[    35  25000]
[    28 123000]
[    35  73000]
[    28  37000]
[    27  88000]
[    28  59000]
[    32  86000]
[    33 149000]
[    19  21000]
[    21  72000]
[    26  35000]
[    27  89000]
[    26  86000]
[    38  80000]
[    39  71000]
[    37  71000]
[    38  61000]
[    37  55000]
[    42  80000]
[    40  57000]
[    35  75000]
[    36  52000]
[    40  59000]
[    41  59000]
```

```
[    36  75000]
[    37  72000]
[    40  75000]
[    35  53000]
[    41  51000]
[    39  61000]
[    42  65000]
[    26  32000]
[    30  17000]
[    26  84000]
[    31  58000]
[    33  31000]
[    30  87000]
[    21  68000]
[    28  55000]
[    23  63000]
[    20  82000]
[    30 107000]
[    28  59000]
[    19  25000]
[    19  85000]
[    18  68000]
[    35  59000]
[    30  89000]
[    34  25000]
[    24  89000]
[    27  96000]
[    41  30000]
[    29  61000]
[    20  74000]
[    26  15000]
[    41  45000]
[    31  76000]
[    36  50000]
[    40  47000]
[    31  15000]
[    46  59000]
[    29  75000]
[    26  30000]
[    32 135000]
[    32 100000]
[    25  90000]
[    37  33000]
[    35  38000]
```

```
[    33  69000]
[    18  86000]
[    22  55000]
[    35  71000]
[    29 148000]
[    29  47000]
[    21  88000]
[    34 115000]
[    26 118000]
[    34  43000]
[    34  72000]
[    23  28000]
[    35  47000]
[    25  22000]
[    24  23000]
[    31  34000]
[    26  16000]
[    31  71000]
[    32 117000]
[    33  43000]
[    33  60000]
[    31  66000]
[    20  82000]
[    33  41000]
[    35  72000]
[    28  32000]
[    24  84000]
[    19  26000]
[    29  43000]
[    19  70000]
[    28  89000]
[    34  43000]
[    30  79000]
[    20  36000]
[    26  80000]
[    35  22000]
[    35  39000]
[    49  74000]
[    39 134000]
[    41  71000]
[    58 101000]
[    47  47000]
[    55 130000]
[    52 114000]
```

```
[    40 142000]
[    46  22000]
[    48  96000]
[    52 150000]
[    59  42000]
[    35  58000]
[    47  43000]
[    60 108000]
[    49  65000]
[    40  78000]
[    46  96000]
[    59 143000]
[    41  80000]
[    35  91000]
[    37 144000]
[    60 102000]
[    35  60000]
[    37  53000]
[    36 126000]
[    56 133000]
[    40  72000]
[    42  80000]
[    35 147000]
[    39  42000]
[    40 107000]
[    49  86000]
[    38 112000]
[    46  79000]
[    40  57000]
[    37  80000]
[    46  82000]
[    53 143000]
[    42 149000]
[    38  59000]
[    50  88000]
[    56 104000]
[    41  72000]
[    51 146000]
[    35  50000]
[    57 122000]
[    41  52000]
[    35  97000]
[    44  39000]
[    37  52000]
```

```
[    48 134000]
[    37 146000]
[    50  44000]
[    52  90000]
[    41  72000]
[    40  57000]
[    58  95000]
[    45 131000]
[    35  77000]
[    36 144000]
[    55 125000]
[    35  72000]
[    48  90000]
[    42 108000]
[    40  75000]
[    37  74000]
[    47 144000]
[    40  61000]
[    43 133000]
[    59  76000]
[    60  42000]
[    39 106000]
[    57  26000]
[    57  74000]
[    38  71000]
[    49  88000]
[    52  38000]
[    50  36000]
[    59  88000]
[    35  61000]
[    37  70000]
[    52  21000]
[    48 141000]
[    37  93000]
[    37  62000]
[    48 138000]
[    41  79000]
[    37  78000]
[    39 134000]
[    49  89000]
[    55  39000]
[    37  77000]
[    35  57000]
[    36  63000]
```

```
[    42  73000]
[    43 112000]
[    45  79000]
[    46 117000]
[    58  38000]
[    48  74000]
[    37 137000]
[    37  79000]
[    40  60000]
[    42  54000]
[    51 134000]
[    47 113000]
[    36 125000]
[    38  50000]
[    42  70000]
[    39  96000]
[    38  50000]
[    49 141000]
[    39  79000]
[    39  75000]
[    54 104000]
[    35  55000]
[    45  32000]
[    36  60000]
[    52 138000]
[    53  82000]
[    41  52000]
[    48  30000]
[    48 131000]
[    41  60000]
[    41  72000]
[    42  75000]
[    36 118000]
[    47 107000]
[    38  51000]
[    48 119000]
[    42  65000]
[    40  65000]
[    57  60000]
[    36  54000]
[    58 144000]
[    35  79000]
[    38  55000]
[    39 122000]
```

```
[    53 104000]
[    35  75000]
[    38  65000]
[    47  51000]
[    47 105000]
[    41  63000]
[    53  72000]
[    54 108000]
[    39  77000]
[    38  61000]
[    38 113000]
[    37  75000]
[    42  90000]
[    37  57000]
[    36  99000]
[    60  34000]
[    54  70000]
[    41  72000]
[    40  71000]
[    42  54000]
[    43 129000]
[    53  34000]
[    47  50000]
[    42  79000]
[    42 104000]
[    59  29000]
[    58  47000]
[    46  88000]
[    38  71000]
[    54  26000]
[    60  46000]
[    60  83000]
[    39  73000]
[    59 130000]
[    37  80000]
[    46  32000]
[    46  74000]
[    42  53000]
[    41  87000]
[    58  23000]
[    42  64000]
[    48  33000]
[    44 139000]
[    49  28000]
```

```
[    57  33000]
[    56  60000]
[    49  39000]
[    39  71000]
[    47  34000]
[    48  35000]
[    48  33000]
[    47  23000]
[    45  45000]
[    60  42000]
[    39  59000]
[    46  41000]
[    51  23000]
[    50  20000]
[    36  33000]
[    49  36000]]
```

In [8]:

```
#Extracting dependent variable:
y = SNA_df.iloc[:, 2].values #Extract kolom terakhir
print(y)
```

```
[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0
 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0
 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1 0 1 1 0 0 0 1 0 0 0 1 0 1
 1 1 0 0 1 1 0 1 1 0 1 1 0 1 0 0 0 1 1 0 1 1 0 1 0 1 0 1 0 0 1 1 0 1 0 0 1
 1 0 1 1 0 1 1 0 0 1 0 0 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1 0 1 1 1 1 0 0 0
 1 1 0 1 1 1 1 1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 1 0 1 1 0 1 1 0 0 0 1 1 0 1 0
 0 1 0 1 0 0 1 1 0 0 1 1 0 1 1 0 0 1 0 1 0 1 1 1 0 1 0 1 1 1 0 1 1 1 1 0 1
 1 1 0 1 0 1 0 0 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1]
```

Feature Scaling

In [9]:

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X = sc.fit_transform(x)
X
```

Out[9]:

```
array([[-1.78179743, -1.49004624],
       [-0.25358736, -1.46068138],
```

```
       [-1.11320552, -0.78528968],
       [-1.01769239, -0.37418169],
       [-1.78179743,  0.18375059],
       [-1.01769239, -0.34481683],
       [-1.01769239,  0.41866944],
       [-0.54012675,  2.35674998],
       [-1.20871865, -1.07893824],
       [-0.25358736, -0.13926283],
       [-1.11320552,  0.30121002],
       [-1.11320552, -0.52100597],
       [-1.6862843 ,  0.47739916],
       [-0.54012675, -1.51941109],
       [-1.87731056,  0.35993973],
       [-0.82666613,  0.30121002],
       [ 0.89257019, -1.3138571 ],
       [ 0.70154394, -1.28449224],
       [ 0.79705706, -1.22576253],
       [ 0.98808332, -1.19639767],
       [ 0.70154394, -1.40195167],
       [ 0.89257019, -0.60910054],
       [ 0.98808332, -0.84401939],
       [ 0.70154394, -1.40195167],
       [ 0.79705706, -1.37258681],
       [ 0.89257019, -1.46068138],
       [ 1.08359645, -1.22576253],
       [ 0.89257019, -1.16703281],
       [-0.82666613, -0.78528968],
       [-0.63563988, -1.51941109],
       [-0.63563988,  0.12502088],
       [-1.01769239,  1.97500684],
       [-1.59077117, -1.5781408 ],
       [-0.92217926, -0.75592482],
       [-1.01769239,  0.59485858],
       [-0.25358736, -1.25512738],
       [-0.44461362, -1.22576253],
       [-0.73115301, -0.60910054],
       [-1.11320552,  0.06629116],
       [-1.01769239, -1.13766796],
       [-1.01769239, -1.54877595],
       [-0.44461362, -0.55037082],
       [-0.25358736,  1.123426  ],
       [-0.73115301, -1.60750566],
       [-0.92217926,  0.41866944],
       [-1.39974491, -1.46068138],
```

```
       [-1.20871865,  0.27184516],
       [-1.01769239, -0.46227625],
       [-0.73115301,  1.91627713],
       [-0.63563988,  0.56549373],
       [-1.30423178, -1.1083031 ],
       [-1.87731056, -0.75592482],
       [-0.82666613,  0.38930459],
       [-0.25358736, -1.37258681],
       [-1.01769239, -0.34481683],
       [-1.30423178, -0.4329114 ],
       [-1.39974491, -0.63846539],
       [-0.92217926,  0.27184516],
       [-1.49525804, -1.51941109],
       [-0.54012675,  1.38770971],
       [-1.01769239, -1.46068138],
       [-1.20871865,  0.50676401],
       [-1.39974491, -0.10989798],
       [-0.54012675,  1.47580428],
       [ 2.03872775,  0.38930459],
       [-1.30423178, -0.34481683],
       [-1.30423178, -1.49004624],
       [-1.39974491,  0.35993973],
       [-1.49525804, -0.19799255],
       [-0.63563988, -0.05116826],
       [-1.20871865,  0.30121002],
       [-1.30423178, -1.25512738],
       [-1.6862843 , -1.37258681],
       [-0.44461362,  1.27025028],
       [-0.54012675, -1.51941109],
       [-0.34910049,  1.24088543],
       [-1.87731056, -0.52100597],
       [-1.49525804, -1.25512738],
       [-0.92217926,  0.50676401],
       [-1.11320552, -1.54877595],
       [-0.73115301,  0.30121002],
       [ 0.12846516, -0.81465453],
       [-1.6862843 , -0.60910054],
       [-0.25358736,  0.53612887],
       [-0.73115301, -0.2273574 ],
       [-0.63563988,  1.41707457],
       [-1.30423178, -0.4329114 ],
       [-0.92217926,  0.4480343 ],
       [-1.11320552,  0.33057487],
       [-0.25358736, -0.57973568],
```

```
       [-1.49525804,  0.33057487],
       [-0.73115301,  1.35834485],
       [-1.11320552, -1.60750566],
       [-0.82666613, -1.22576253],
       [-0.82666613,  0.38930459],
       [-0.25358736, -0.75592482],
       [-0.25358736, -1.3138571 ],
       [-0.92217926,  1.56389885],
       [-0.25358736,  0.09565602],
       [-0.92217926, -0.96147882],
       [-1.01769239,  0.53612887],
       [-0.92217926, -0.31545197],
       [-0.54012675,  0.47739916],
       [-0.44461362,  2.32738512],
       [-1.78179743, -1.43131652],
       [-1.59077117,  0.06629116],
       [-1.11320552, -1.02020853],
       [-1.01769239,  0.56549373],
       [-1.11320552,  0.47739916],
       [ 0.03295203,  0.30121002],
       [ 0.12846516,  0.03692631],
       [-0.0625611 ,  0.03692631],
       [ 0.03295203, -0.25672226],
       [-0.0625611 , -0.4329114 ],
       [ 0.41500455,  0.30121002],
       [ 0.22397829, -0.37418169],
       [-0.25358736,  0.15438573],
       [-0.15807423, -0.52100597],
       [ 0.22397829, -0.31545197],
       [ 0.31949142, -0.31545197],
       [-0.15807423,  0.15438573],
       [-0.0625611 ,  0.06629116],
       [ 0.22397829,  0.15438573],
       [-0.25358736, -0.49164111],
       [ 0.31949142, -0.55037082],
       [ 0.12846516, -0.25672226],
       [ 0.41500455, -0.13926283],
       [-1.11320552, -1.1083031 ],
       [-0.73115301, -1.54877595],
       [-1.11320552,  0.41866944],
       [-0.63563988, -0.34481683],
       [-0.44461362, -1.13766796],
       [-0.73115301,  0.50676401],
       [-1.59077117, -0.05116826],
```

```
       [-0.92217926, -0.4329114 ],
       [-1.39974491, -0.19799255],
       [-1.6862843 ,  0.35993973],
       [-0.73115301,  1.09406114],
       [-0.92217926, -0.31545197],
       [-1.78179743, -1.3138571 ],
       [-1.78179743,  0.4480343 ],
       [-1.87731056, -0.05116826],
       [-0.25358736, -0.31545197],
       [-0.73115301,  0.56549373],
       [-0.34910049, -1.3138571 ],
       [-1.30423178,  0.56549373],
       [-1.01769239,  0.77104772],
       [ 0.31949142, -1.16703281],
       [-0.82666613, -0.25672226],
       [-1.6862843 ,  0.12502088],
       [-1.11320552, -1.60750566],
       [ 0.31949142, -0.72655996],
       [-0.63563988,  0.18375059],
       [-0.15807423, -0.57973568],
       [ 0.22397829, -0.66783025],
       [-0.63563988, -1.60750566],
       [ 0.79705706, -0.31545197],
       [-0.82666613,  0.15438573],
       [-1.11320552, -1.16703281],
       [-0.54012675,  1.91627713],
       [-0.54012675,  0.88850715],
       [-1.20871865,  0.59485858],
       [-0.0625611 , -1.07893824],
       [-0.25358736, -0.93211396],
       [-0.44461362, -0.02180341],
       [-1.87731056,  0.47739916],
       [-1.49525804, -0.4329114 ],
       [-0.25358736,  0.03692631],
       [-0.82666613,  2.29802026],
       [-0.82666613, -0.66783025],
       [-1.59077117,  0.53612887],
       [-0.34910049,  1.32898   ],
       [-1.11320552,  1.41707457],
       [-0.34910049, -0.78528968],
       [-0.34910049,  0.06629116],
       [-1.39974491, -1.22576253],
       [-0.25358736, -0.66783025],
       [-1.20871865, -1.40195167],
```

```
       [-1.30423178, -1.37258681],
       [-0.63563988, -1.04957339],
       [-1.11320552, -1.5781408 ],
       [-0.63563988,  0.03692631],
       [-0.54012675,  1.38770971],
       [-0.44461362, -0.78528968],
       [-0.44461362, -0.28608712],
       [-0.63563988, -0.10989798],
       [-1.6862843 ,  0.35993973],
       [-0.44461362, -0.84401939],
       [-0.25358736,  0.06629116],
       [-0.92217926, -1.1083031 ],
       [-1.30423178,  0.41866944],
       [-1.78179743, -1.28449224],
       [-0.82666613, -0.78528968],
       [-1.78179743,  0.00756145],
       [-0.92217926,  0.56549373],
       [-0.34910049, -0.78528968],
       [-0.73115301,  0.27184516],
       [-1.6862843 , -0.99084367],
       [-1.11320552,  0.30121002],
       [-0.25358736, -1.40195167],
       [-0.25358736, -0.9027491 ],
       [ 1.08359645,  0.12502088],
       [ 0.12846516,  1.88691227],
       [ 0.31949142,  0.03692631],
       [ 1.94321462,  0.917872  ],
       [ 0.89257019, -0.66783025],
       [ 1.65667523,  1.76945285],
       [ 1.37013584,  1.29961514],
       [ 0.22397829,  2.12183112],
       [ 0.79705706, -1.40195167],
       [ 0.98808332,  0.77104772],
       [ 1.37013584,  2.35674998],
       [ 2.03872775, -0.81465453],
       [-0.25358736, -0.34481683],
       [ 0.89257019, -0.78528968],
       [ 2.13424088,  1.123426  ],
       [ 1.08359645, -0.13926283],
       [ 0.22397829,  0.2424803 ],
       [ 0.79705706,  0.77104772],
       [ 2.03872775,  2.15119598],
       [ 0.31949142,  0.30121002],
       [-0.25358736,  0.62422344],
```

```
       [-0.0625611 ,  2.18056084],
       [ 2.13424088,  0.94723686],
       [-0.25358736, -0.28608712],
       [-0.0625611 , -0.49164111],
       [-0.15807423,  1.65199342],
       [ 1.75218836,  1.85754742],
       [ 0.22397829,  0.06629116],
       [ 0.41500455,  0.30121002],
       [-0.25358736,  2.26865541],
       [ 0.12846516, -0.81465453],
       [ 0.22397829,  1.09406114],
       [ 1.08359645,  0.47739916],
       [ 0.03295203,  1.24088543],
       [ 0.79705706,  0.27184516],
       [ 0.22397829, -0.37418169],
       [-0.0625611 ,  0.30121002],
       [ 0.79705706,  0.35993973],
       [ 1.46564897,  2.15119598],
       [ 0.41500455,  2.32738512],
       [ 0.03295203, -0.31545197],
       [ 1.17910958,  0.53612887],
       [ 1.75218836,  1.00596657],
       [ 0.31949142,  0.06629116],
       [ 1.27462271,  2.23929055],
       [-0.25358736, -0.57973568],
       [ 1.84770149,  1.53453399],
       [ 0.31949142, -0.52100597],
       [-0.25358736,  0.80041258],
       [ 0.60603081, -0.9027491 ],
       [-0.0625611 , -0.52100597],
       [ 0.98808332,  1.88691227],
       [-0.0625611 ,  2.23929055],
       [ 1.17910958, -0.75592482],
       [ 1.37013584,  0.59485858],
       [ 0.31949142,  0.06629116],
       [ 0.22397829, -0.37418169],
       [ 1.94321462,  0.74168287],
       [ 0.70154394,  1.7988177 ],
       [-0.25358736,  0.21311545],
       [-0.15807423,  2.18056084],
       [ 1.65667523,  1.62262856],
       [-0.25358736,  0.06629116],
       [ 0.98808332,  0.59485858],
       [ 0.41500455,  1.123426  ],
```

```
       [ 0.22397829,  0.15438573],
       [-0.0625611 ,  0.12502088],
       [ 0.89257019,  2.18056084],
       [ 0.22397829, -0.25672226],
       [ 0.51051768,  1.85754742],
       [ 2.03872775,  0.18375059],
       [ 2.13424088, -0.81465453],
       [ 0.12846516,  1.06469629],
       [ 1.84770149, -1.28449224],
       [ 1.84770149,  0.12502088],
       [ 0.03295203,  0.03692631],
       [ 1.08359645,  0.53612887],
       [ 1.37013584, -0.93211396],
       [ 1.17910958, -0.99084367],
       [ 2.03872775,  0.53612887],
       [-0.25358736, -0.25672226],
       [-0.0625611 ,  0.00756145],
       [ 1.37013584, -1.43131652],
       [ 0.98808332,  2.09246627],
       [-0.0625611 ,  0.68295315],
       [-0.0625611 , -0.2273574 ],
       [ 0.98808332,  2.0043717 ],
       [ 0.31949142,  0.27184516],
       [-0.0625611 ,  0.2424803 ],
       [ 0.12846516,  1.88691227],
       [ 1.08359645,  0.56549373],
       [ 1.65667523, -0.9027491 ],
       [-0.0625611 ,  0.21311545],
       [-0.25358736, -0.37418169],
       [-0.15807423, -0.19799255],
       [ 0.41500455,  0.09565602],
       [ 0.51051768,  1.24088543],
       [ 0.70154394,  0.27184516],
       [ 0.79705706,  1.38770971],
       [ 1.94321462, -0.93211396],
       [ 0.98808332,  0.12502088],
       [-0.0625611 ,  1.97500684],
       [-0.0625611 ,  0.27184516],
       [ 0.22397829, -0.28608712],
       [ 0.41500455, -0.46227625],
       [ 1.27462271,  1.88691227],
       [ 0.89257019,  1.27025028],
       [-0.15807423,  1.62262856],
       [ 0.03295203, -0.57973568],
```

```
       [ 0.41500455,  0.00756145],
       [ 0.12846516,  0.77104772],
       [ 0.03295203, -0.57973568],
       [ 1.08359645,  2.09246627],
       [ 0.12846516,  0.27184516],
       [ 0.12846516,  0.15438573],
       [ 1.5611621 ,  1.00596657],
       [-0.25358736, -0.4329114 ],
       [ 0.70154394, -1.1083031 ],
       [-0.15807423, -0.28608712],
       [ 1.37013584,  2.0043717 ],
       [ 1.46564897,  0.35993973],
       [ 0.31949142, -0.52100597],
       [ 0.98808332, -1.16703281],
       [ 0.98808332,  1.7988177 ],
       [ 0.31949142, -0.28608712],
       [ 0.31949142,  0.06629116],
       [ 0.41500455,  0.15438573],
       [-0.15807423,  1.41707457],
       [ 0.89257019,  1.09406114],
       [ 0.03295203, -0.55037082],
       [ 0.98808332,  1.44643942],
       [ 0.41500455, -0.13926283],
       [ 0.22397829, -0.13926283],
       [ 1.84770149, -0.28608712],
       [-0.15807423, -0.46227625],
       [ 1.94321462,  2.18056084],
       [-0.25358736,  0.27184516],
       [ 0.03295203, -0.4329114 ],
       [ 0.12846516,  1.53453399],
       [ 1.46564897,  1.00596657],
       [-0.25358736,  0.15438573],
       [ 0.03295203, -0.13926283],
       [ 0.89257019, -0.55037082],
       [ 0.89257019,  1.03533143],
       [ 0.31949142, -0.19799255],
       [ 1.46564897,  0.06629116],
       [ 1.5611621 ,  1.123426  ],
       [ 0.12846516,  0.21311545],
       [ 0.03295203, -0.25672226],
       [ 0.03295203,  1.27025028],
       [-0.0625611 ,  0.15438573],
       [ 0.41500455,  0.59485858],
       [-0.0625611 , -0.37418169],
```

```
       [-0.15807423,  0.85914229],
       [ 2.13424088, -1.04957339],
       [ 1.5611621 ,  0.00756145],
       [ 0.31949142,  0.06629116],
       [ 0.22397829,  0.03692631],
       [ 0.41500455, -0.46227625],
       [ 0.51051768,  1.74008799],
       [ 1.46564897, -1.04957339],
       [ 0.89257019, -0.57973568],
       [ 0.41500455,  0.27184516],
       [ 0.41500455,  1.00596657],
       [ 2.03872775, -1.19639767],
       [ 1.94321462, -0.66783025],
       [ 0.79705706,  0.53612887],
       [ 0.03295203,  0.03692631],
       [ 1.5611621 , -1.28449224],
       [ 2.13424088, -0.69719511],
       [ 2.13424088,  0.38930459],
       [ 0.12846516,  0.09565602],
       [ 2.03872775,  1.76945285],
       [-0.0625611 ,  0.30121002],
       [ 0.79705706, -1.1083031 ],
       [ 0.79705706,  0.12502088],
       [ 0.41500455, -0.49164111],
       [ 0.31949142,  0.50676401],
       [ 1.94321462, -1.37258681],
       [ 0.41500455, -0.16862769],
       [ 0.98808332, -1.07893824],
       [ 0.60603081,  2.03373655],
       [ 1.08359645, -1.22576253],
       [ 1.84770149, -1.07893824],
       [ 1.75218836, -0.28608712],
       [ 1.08359645, -0.9027491 ],
       [ 0.12846516,  0.03692631],
       [ 0.89257019, -1.04957339],
       [ 0.98808332, -1.02020853],
       [ 0.98808332, -1.07893824],
       [ 0.89257019, -1.37258681],
       [ 0.70154394, -0.72655996],
       [ 2.13424088, -0.81465453],
       [ 0.12846516, -0.31545197],
       [ 0.79705706, -0.84401939],
       [ 1.27462271, -1.37258681],
       [ 1.17910958, -1.46068138],
```

```
[-0.15807423, -1.07893824],
[ 1.08359645, -0.99084367]])
```

Splitting dataset into Training set and Test set

With 400 data, I decided to split the dataset into 3:1 ratio (Training set contains 300 data, Test set contains 100 data)

In [10]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)
print('Train set: ', X_train.shape, y_train.shape)
print('Test set: ', X_test.shape, y_test.shape)
```

```
Train set:  (300, 2) (300,)
Test set:  (100, 2) (100,)
```

Create classifier using Logistic Regression

In [11]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LR = LogisticRegression(C=0.01, solver = 'liblinear').fit(X_train, y_train)
LR
```

Out[11]:
```
LogisticRegression(C=0.01, solver='liblinear')
```

Predict Purchased (1) or Not Purchased (0)

In [12]:
```
#Predict using test set
yhat = LR.predict(X_test)
yhat
```

Out[12]:
```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1,
       0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1], dtype=int64)
```

In [13]:
```
#Predict probability for each class
yhat_prob = LR.predict_proba(X_test)
yhat_prob
```

Out[13]:
```
array([[0.61071527, 0.38928473],
       [0.59481337, 0.40518663],
       [0.57875015, 0.42124985],
       [0.62618269, 0.37381731],
       [0.62773323, 0.37226677],
       [0.75730282, 0.24269718],
       [0.72989984, 0.27010016],
       [0.42703227, 0.57296773],
       [0.76141118, 0.23858882],
       [0.50636133, 0.49363867],
       [0.68172416, 0.31827584],
       [0.69103094, 0.30896906],
       [0.59607312, 0.40392688],
       [0.52907959, 0.47092041],
       [0.71515003, 0.28484997],
       [0.52799194, 0.47200806],
       [0.55202425, 0.44797575],
       [0.7261002 , 0.2738998 ],
       [0.25786828, 0.74213172],
       [0.67960291, 0.32039709],
       [0.63345966, 0.36654034],
       [0.30955792, 0.69044208],
       [0.55899908, 0.44100092],
       [0.38163704, 0.61836296],
       [0.77854369, 0.22145631],
       [0.29525288, 0.70474712],
       [0.63244561, 0.36755439],
       [0.6350358 , 0.3649642 ],
       [0.58218968, 0.41781032],
       [0.59051895, 0.40948105],
       [0.70222995, 0.29777005],
       [0.54796804, 0.45203196],
       [0.35988265, 0.64011735],
       [0.60001276, 0.39998724],
       [0.72415503, 0.27584497],
       [0.7844081 , 0.2155919 ],
       [0.7045581 , 0.2954419 ],
       [0.6515971 , 0.3484029 ],
       [0.69813256, 0.30186744],
       [0.4976011 , 0.5023989 ],
       [0.64123538, 0.35876462],
       [0.55697804, 0.44302196],
       [0.64874053, 0.35125947],
       [0.68316089, 0.31683911],
```

```
       [0.41213166, 0.58786834],
       [0.69562746, 0.30437254],
       [0.54593751, 0.45406249],
       [0.35647483, 0.64352517],
       [0.75067328, 0.24932672],
       [0.39668061, 0.60331939],
       [0.25276382, 0.74723618],
       [0.68429176, 0.31570824],
       [0.61423372, 0.38576628],
       [0.50901668, 0.49098332],
       [0.28951935, 0.71048065],
       [0.55181038, 0.44818962],
       [0.62931958, 0.37068042],
       [0.66651989, 0.33348011],
       [0.51398621, 0.48601379],
       [0.78963969, 0.21036031],
       [0.71572008, 0.28427992],
       [0.36193417, 0.63806583],
       [0.74052042, 0.25947958],
       [0.51868574, 0.48131426],
       [0.81128704, 0.18871296],
       [0.28844482, 0.71155518],
       [0.67065308, 0.32934692],
       [0.69562746, 0.30437254],
       [0.57836861, 0.42163139],
       [0.50880056, 0.49119944],
       [0.46049999, 0.53950001],
       [0.57474952, 0.42525048],
       [0.73430933, 0.26569067],
       [0.57197949, 0.42802051],
       [0.63934807, 0.36065193],
       [0.73995033, 0.26004967],
       [0.49036528, 0.50963472],
       [0.55219735, 0.44780265],
       [0.45561387, 0.54438613],
       [0.37777702, 0.62222298],
       [0.21241925, 0.78758075],
       [0.29801931, 0.70198069],
       [0.72903847, 0.27096153],
       [0.74292637, 0.25707363],
       [0.39131252, 0.60868748],
       [0.48831761, 0.51168239],
       [0.52164571, 0.47835429],
       [0.22410821, 0.77589179],
```

```
       [0.52199505, 0.47800495],
       [0.54074726, 0.45925274],
       [0.52108055, 0.47891945],
       [0.41179251, 0.58820749],
       [0.74900487, 0.25099513],
       [0.75469747, 0.24530253],
       [0.67963878, 0.32036122],
       [0.62601882, 0.37398118],
       [0.72519844, 0.27480156],
       [0.51529249, 0.48470751],
       [0.39989772, 0.60010228],
       [0.43944779, 0.56055221]])
```

Visualising the Training and Test set results

In [14]:

```
from matplotlib.colors import ListedColormap

x1, x2 = np.meshgrid(np.arange(start = X_train[:, 0].min()-1, stop = X_train[:, 0].max()+1, step = 0.01),
                     np.arange(start = X_train[:, 1].min()-1, stop = X_train[:, 1].max()+1, step = 0.01))

plt.contourf(x1, x2, LR.predict(np.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
                                alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(x1.min(), x1.max())
plt.ylim(x2.min(), x2.max())

for i, j in enumerate(np.unique(y_train)):
    plt.scatter(X_train[y_train == j, 0], X_train[y_train == j, 1],
    c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Logistic Regression (Training Set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')

plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in ca
se its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a single row if you inte
nd to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in ca
```

```
se its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a single row if you inte
nd to specify the same RGB or RGBA value for all points.
```

In [15]:

```
x1, x2 = np.meshgrid(np.arange(start = X_test[:, 0].min()-1, stop = X_test[:, 0].max()+1, step = 0.01),
                     np.arange(start = X_test[:, 1].min()-1, stop = X_test[:, 1].max()+1, step = 0.01))

plt.contourf(x1, x2, LR.predict(np.array([x1.ravel(), x2.ravel()]).T).reshape(x1.shape),
                                alpha = 0.75, cmap = ListedColormap(('red', 'green')))

plt.xlim(x1.min(), x1.max())
plt.ylim(x2.min(), x2.max())

for i, j in enumerate(np.unique(y_test)):
    plt.scatter(X_test[y_test == j, 0], X_test[y_test == j, 1],
    c = ListedColormap(('red', 'green'))(i), label = j)

plt.title('Logistic Regression (Test Set)')
plt.xlabel('Age')
plt.ylabel('Estimated Salary')

plt.legend()
plt.show()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in ca
se its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a single row if you inte
```

```
nd to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be avoided as value-mapping will have precedence in ca
se its length matches with *x* & *y*.  Please use the *color* keyword-argument or provide a 2D array with a single row if you inte
nd to specify the same RGB or RGBA value for all points.
```

Evaluation

Jaccard Index

In [16]:
```
from sklearn.metrics import jaccard_score
jaccard_score(y_test, yhat)
```

Out[16]:
```
0.7058823529411765
```

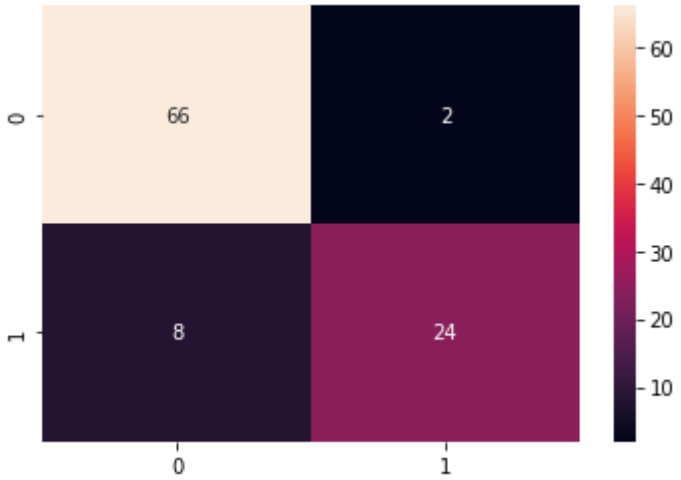
Confusion Matrix

In [17]:
```
from sklearn.metrics import classification_report, confusion_matrix
conf_matrix = confusion_matrix(y_test, yhat)
conf_matrix
```

Out[17]:
```
array([[66,  2],
       [ 8, 24]], dtype=int64)
```

In [18]:
```
sns.heatmap(conf_matrix, annot=True)
```

Out[18]:
```
<AxesSubplot:>
```



Classification Report

In [19]:
```
print(classification_report(y_test, yhat))
```

```
              precision    recall  f1-score   support

           0       0.89      0.97      0.93        68
           1       0.92      0.75      0.83        32

    accuracy                           0.90       100
   macro avg       0.91      0.86      0.88       100
weighted avg       0.90      0.90      0.90       100
```

Average accuracy = average of f1-score = **0.90**

Log Loss

In [20]:
```
from sklearn.metrics import log_loss
log_loss(y_test, yhat_prob)
```

```
0.479816705064194
```

Out[20]: