In [ ]:

```
#LOGISTIC REGRESSION
```

In [7]:

```
#Import the usual libraries for pandas and plotting

import pandas as pd
import seaborn as sb
import numpy as np
import sklearn
from matplotlib import pyplot as plt
```

In [8]:

```
#Read “advertising.csv” file and set it to a data frame called ad_data

ad_data = pd.read_csv('C:/Users/Deep/Desktop/advertising.csv')
```

In [9]:

```
#Check the Head of ad_data

ad_data.head()
```

Out[9]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestamp	Clicked on Ad
0	68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	2016-03-27 00:53:11	0
1	80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	2016-04-04 01:39:02	0
2	69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	2016-03-13 20:35:42	0
3	74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	2016-01-10 02:31:19	0
4	68.37	35	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	2016-06-03 03:36:18	0

In [10]:

```
#Check whether any missing data are there or not

ad_data.isnull().sum()
```

Out[10]:

```
Daily Time Spent on Site    0
Age                         0
Area Income                 0
Daily Internet Usage        0
Ad Topic Line               0
City                        0
Male                        0
Country                     0
Timestamp                   0
Clicked on Ad               0
dtype: int64
```

In [11]:

```
#Calculate several statistical measures on numerical attributes

ad_data.describe()
```

Out[11]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Male	Clicked on Ad
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000
mean	65.000200	36.009000	55000.000080	180.000100	0.481000	0.50000
std	15.853615	8.785562	13414.634022	43.902339	0.499889	0.50025
min	32.600000	19.000000	13996.500000	104.780000	0.000000	0.00000
25%	51.360000	29.000000	47031.802500	138.830000	0.000000	0.00000
50%	68.215000	35.000000	57012.300000	183.130000	0.000000	0.50000
75%	78.547500	42.000000	65470.635000	218.792500	1.000000	1.00000
max	91.430000	61.000000	79484.800000	269.960000	1.000000	1.00000

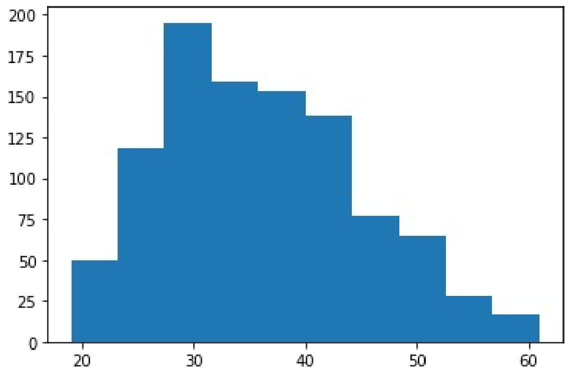
In [21]:

```
#Create a Histogram of the ‘Age’ using Seaborn package

plt.hist(ad_data['Age'])
```

Out[21]:

```
(array([ 50., 118., 195., 159., 153., 138.,  77.,  65.,  28.,  17.]),
 array([19. , 23.2, 27.4, 31.6, 35.8, 40. , 44.2, 48.4, 52.6, 56.8, 61. ]),
 <a list of 10 Patch objects>)
```



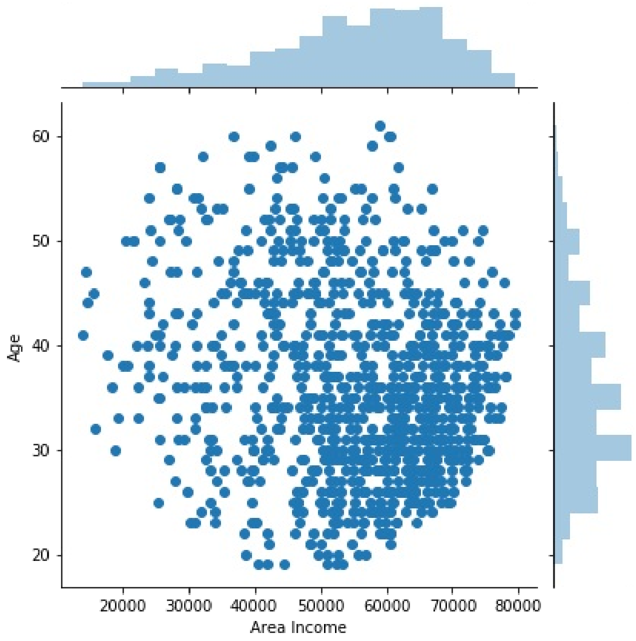
In [22]:

```
#Create a Jointplot showing ‘Area Income’ versus ‘Age’.

sb.jointplot(ad_data['Area Income'], ad_data['Age'])
```

Out[22]:

```
<seaborn.axisgrid.JointGrid at 0x211a66ef248>
```



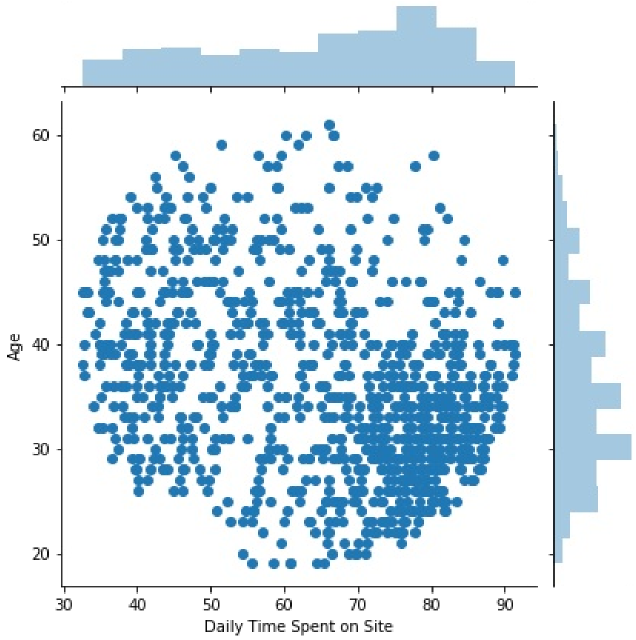
In [23]:

```
# Create a Jointplot showing the ‘kde’ distributions of ‘Daily Time Spent on site’ versus ‘Age’

sb.jointplot(ad_data['Daily Time Spent on Site'], ad_data['Age'])
```

Out[23]:

```
<seaborn.axisgrid.JointGrid at 0x211a670ec08>
```



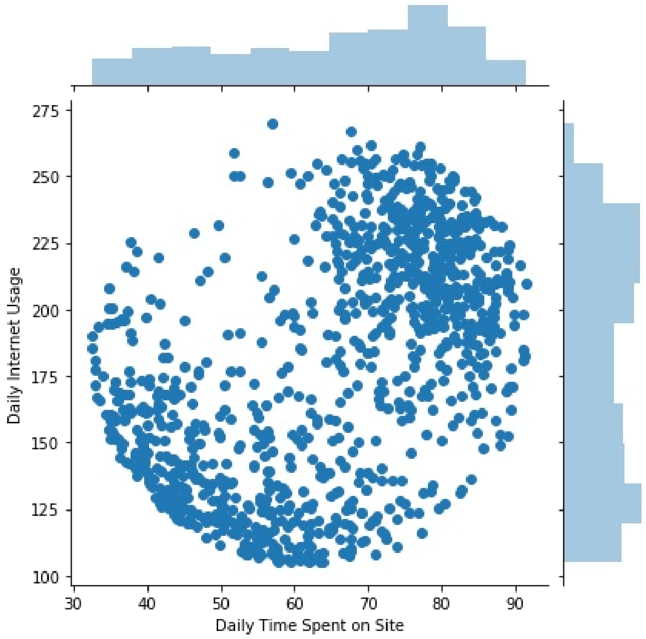
In [24]:

```
#Create a Jointplot of ‘Daily Time Spent on Site’ versus ‘Daily Internet Usage’

sb.jointplot(ad_data['Daily Time Spent on Site'], ad_data['Daily Internet Usage'])
```

Out[24]:

```
<seaborn.axisgrid.JointGrid at 0x211a6e1ae08>
```



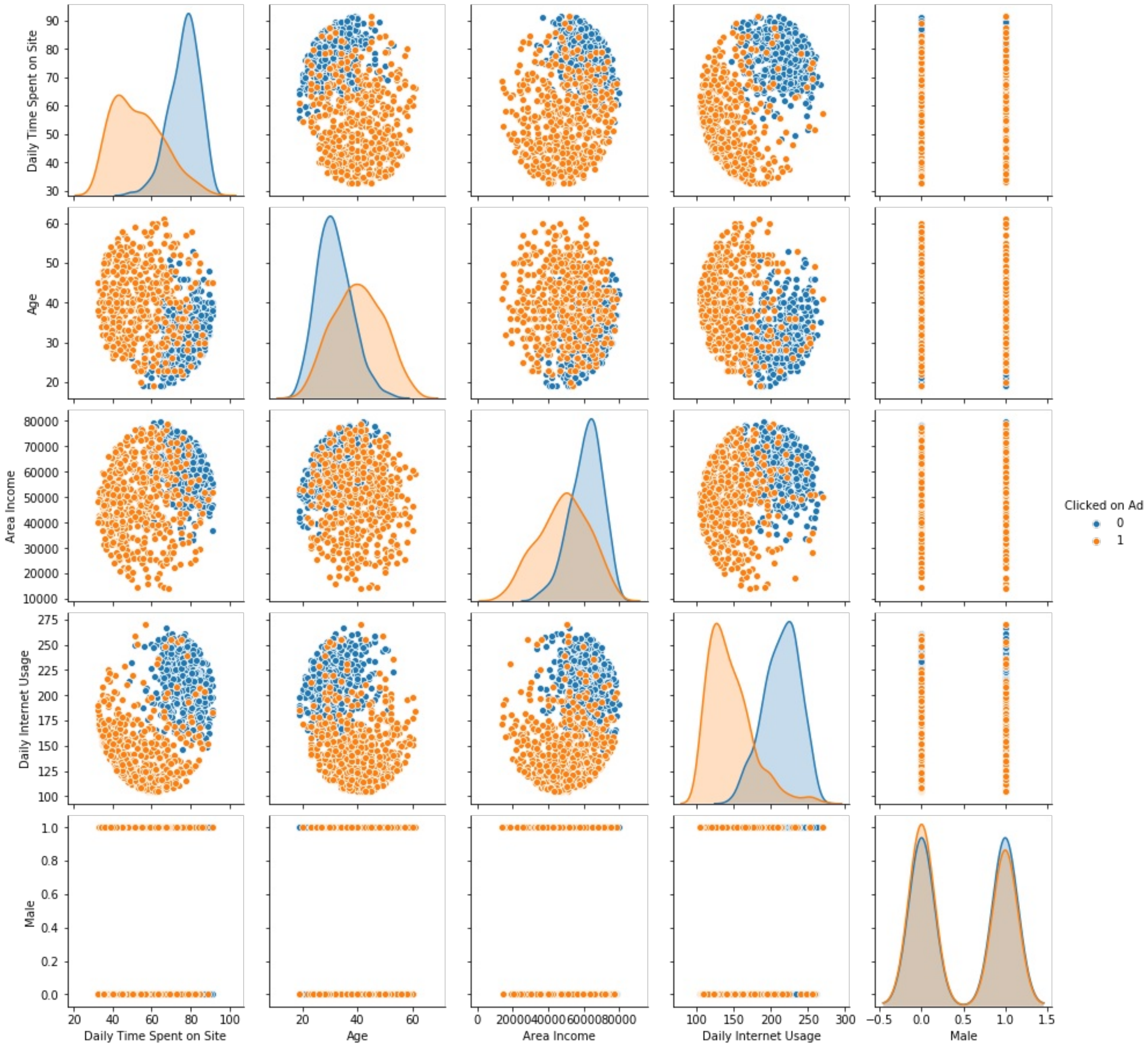
In [25]:

```
#Crate a pairplot with the hue defined by the ‘Clicked on Ad’ column feature

sb.pairplot(ad_data,hue='Clicked on Ad')
```

Out[25]:

```
<seaborn.axisgrid.PairGrid at 0x211a7f2bb88>
```



In [26]:

```
#Choose columns that you want to use for Logistic Regression

x = ad_data[['Daily Time Spent on Site', 'Area Income', 'Daily Internet Usage']]
y = ad_data[['Clicked on Ad']]
```

In [27]:

```
#Split the data into training set and testing set so that your testing set consists 25% of total data

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=42)
```

In [28]:

```
#Train and fit a Logistic Regression model on the training set

from sklearn.linear_model import LogisticRegression
logic = LogisticRegression()
logic.fit(x_train,y_train)
```

```
C:\Users\Deep\anaconda3\lib\site-packages\sklearn\utils\validation.py:760: DataConversionWarning: A
column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_samples,
), for example using ravel().
  y = column_or_1d(y, warn=True)
```

Out[28]:

```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='auto', n_jobs=None, penalty='l2',
                   random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                   warm_start=False)
```

In [29]:

```
#Display all the coefficients values of the fitted Logistic Regression Model

logic.coef_
```

Out[29]:

```
array([[-1.57040367e-01, -1.05229723e-04, -6.98627182e-02]])
```

In [30]:

```
#Predict class label for the testing dataset

predict = logic.predict(x_test)
predict
```

Out[30]:

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

In [31]:

```
#Create a classification report for the model

from sklearn.metrics import classification_report,confusion_matrix
print(classification_report(y_test,predict))
```

```
              precision    recall  f1-score   support

           0       0.92      0.97      0.94       120
           1       0.97      0.92      0.94       130

    accuracy                           0.94       250
   macro avg       0.94      0.94      0.94       250
weighted avg       0.95      0.94      0.94       250
```

In [32]:

```
#Create the confusion matrix for your fitted model and calculate the model accuracy

print(confusion_matrix(y_test,predict))
```

```
[[116   4]
 [ 10 120]]
```