In [1]:

```
import numpy as np
import pandas as pd
```

In [2]:

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:

```
advert = pd.read_csv(r'C:\Users\DELL\Downloads\3PythonCourse\Refactored_Py_DS_ML_Bootcamp-master\13
```

In [4]:

```
advert.head()
```

Out[4]:

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

In [5]:

```
advert.describe()
```

Out[5]:

	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 [6]:

```
advert.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 10 columns):
Daily Time Spent on Site    1000 non-null float64
Age                         1000 non-null int64
Area Income                 1000 non-null float64
Daily Internet Usage        1000 non-null float64
Ad Topic Line               1000 non-null object
City                        1000 non-null object
Male                        1000 non-null int64
Country                     1000 non-null object
Timestamp                   1000 non-null object
Clicked on Ad               1000 non-null int64
dtypes: float64(3), int64(3), object(4)
memory usage: 78.2+ KB
```

In [7]:

```
advert.isnull().sum()
```

Out[7]:

```
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
```

EDA

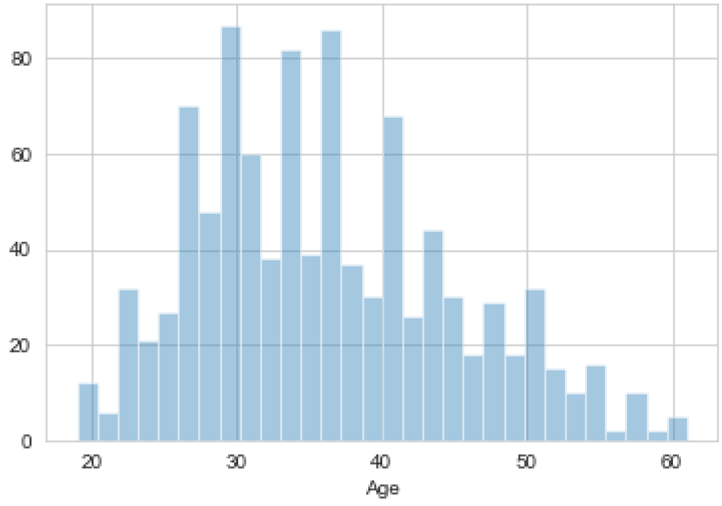
Lets explore the data using seaborn

In [8]:

```
#creating histogram for the Age column
sns.set_style('whitegrid')
sns.distplot(advert['Age'], kde = False, bins = 30)
```

Out[8]:

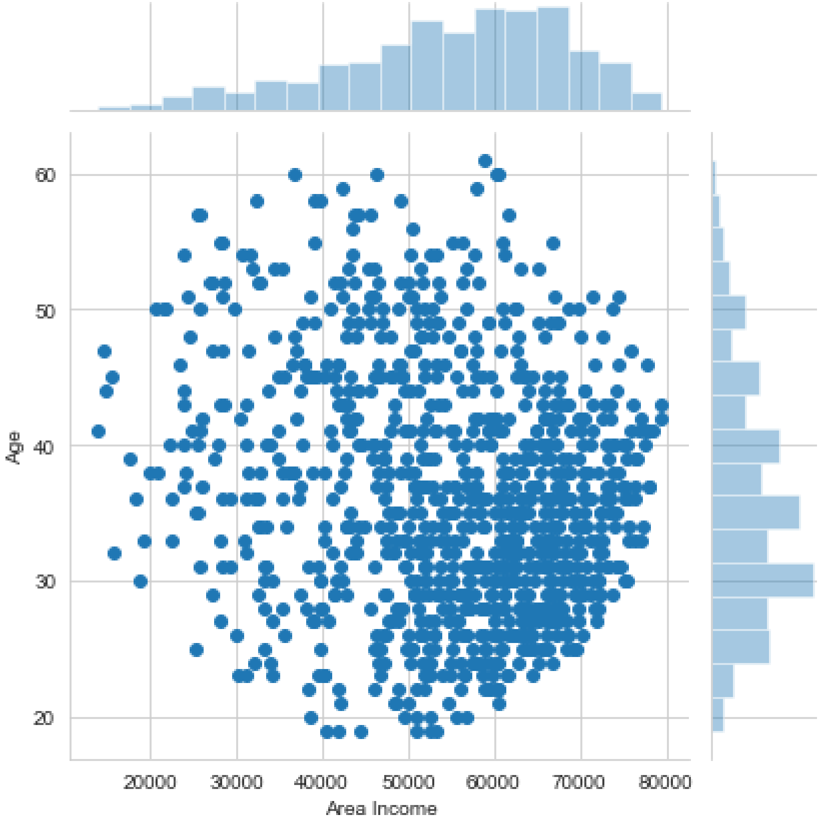
```
<matplotlib.axes._subplots.AxesSubplot at 0x2ed08e80fd0>
```



In [9]:

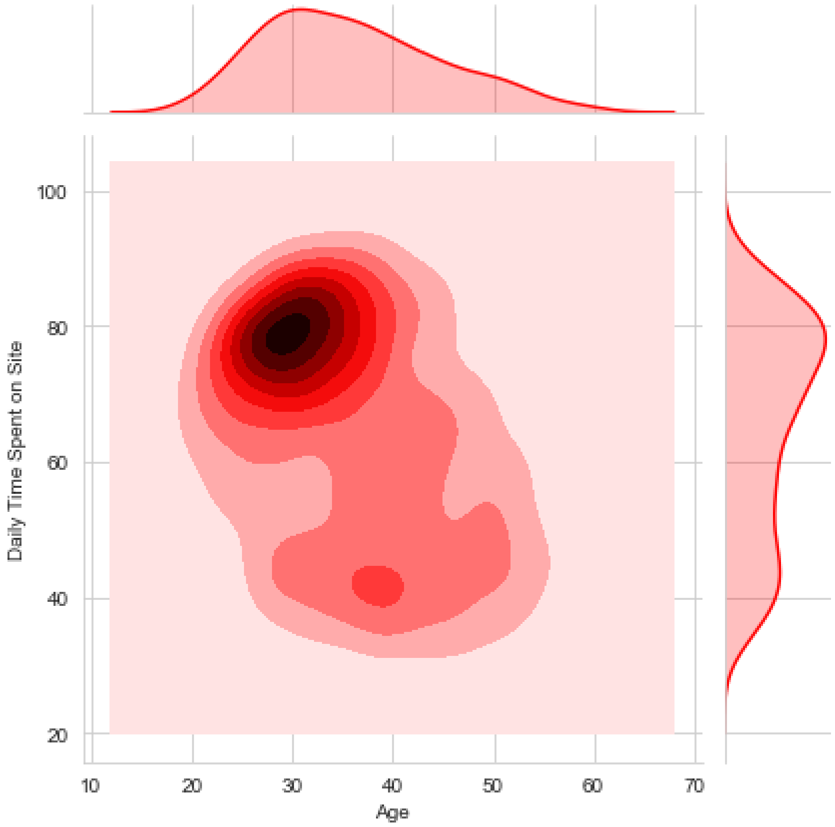
```
sns.jointplot(x = 'Area Income', y = 'Age', data = advert)
plt.show()
```

```
C:\Users\DELL\Anaconda3\lib\site-packages\scipy\stats\stats.py:1713: FutureWarning: Using a
non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of
`arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will r
esult either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



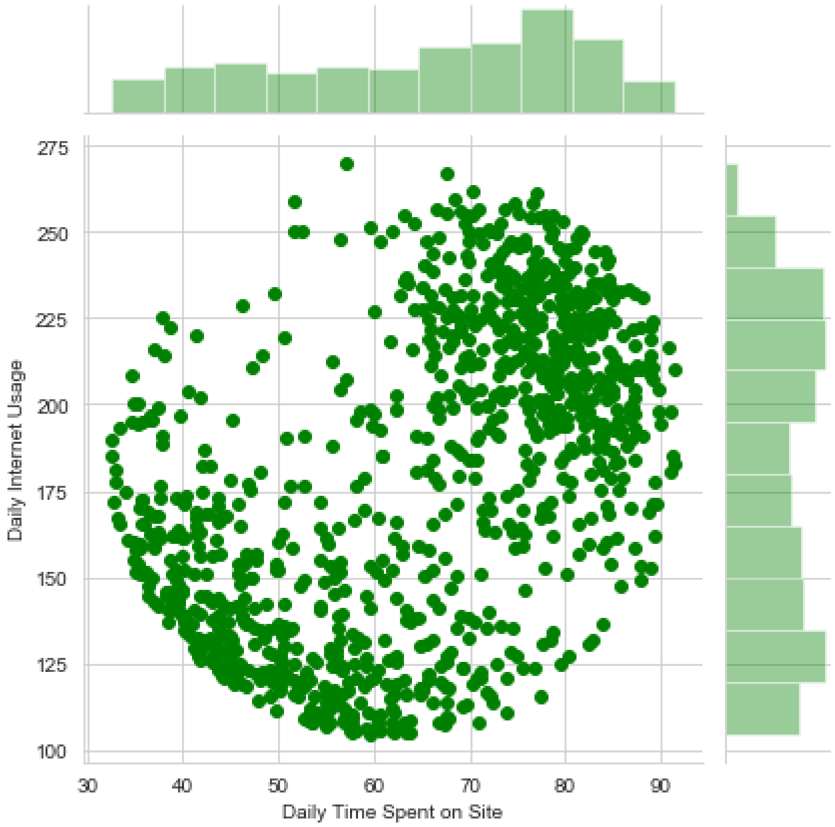
In [10]:

```
sns.jointplot(x =  'Age', y ='Daily Time Spent on Site', data = advert, kind = 'kde', color = 'red')
plt.show()
```



In [11]:

```
sns.jointplot(x =  'Daily Time Spent on Site', y ='Daily Internet Usage', data = advert,  color = 'green')
plt.show()
```



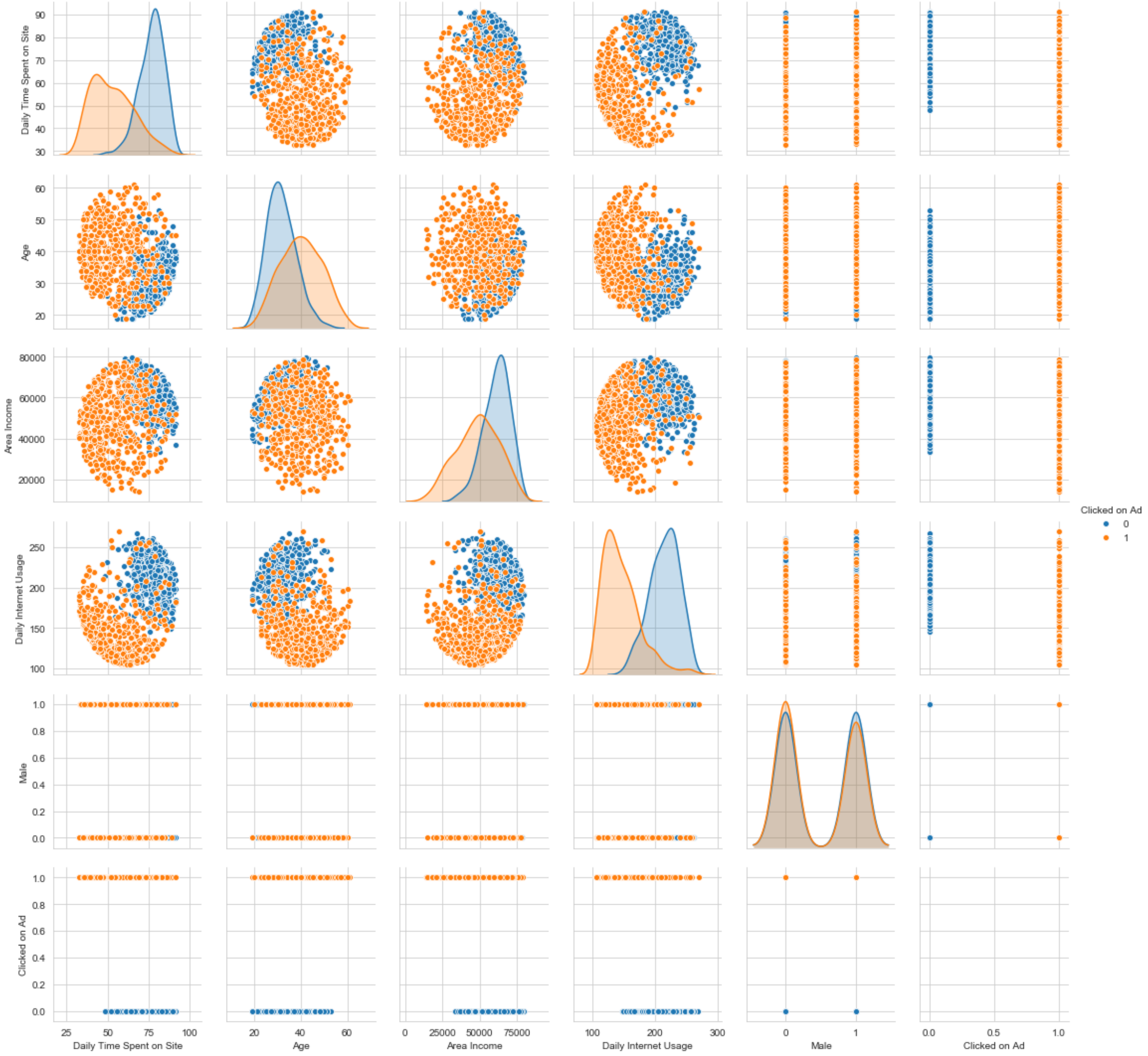
In [12]:

```
sns.pairplot(advert, hue = 'Clicked on Ad')
plt.plot()
```

```
C:\Users\DELL\Anaconda3\lib\site-packages\statsmodels\nonparametric\kde.py:488: Runtime
Warning: invalid value encountered in true_divide
  binned = fast_linbin(X, a, b, gridsize) / (delta * nobs)
C:\Users\DELL\Anaconda3\lib\site-packages\statsmodels\nonparametric\kdetools.py:34: Runti
meWarning: invalid value encountered in double_scalars
  FAC1 = 2*(np.pi*bw/RANGE)**2
C:\Users\DELL\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:83: RuntimeWarning:
invalid value encountered in reduce
  return ufunc.reduce(obj, axis, dtype, out, **passkwargs)
```

Out[12]:

```
[]
```



Logisic Regression

Now since our data is prepared we will now split our data into training and test data

In [22]:

```
advert.columns
```

Out[22]:

```
Index(['Daily Time Spent on Site', 'Age', 'Area Income',
       'Daily Internet Usage', 'Ad Topic Line', 'City', 'Male', 'Country',
       'Timestamp', 'Clicked on Ad'],
      dtype='object')
```

In [23]:

```
X = advert[['Daily Time Spent on Site', 'Age', 'Area Income', 'Daily Internet Usage']]
```

In [24]:

```
y = advert['Clicked on Ad']
```

In [25]:

```
from sklearn.model_selection import train_test_split
```

In [32]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

In [27]:

```
from sklearn.linear_model import LogisticRegression
```

In [28]:

```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
```

```
C:\Users\DELL\Anaconda3\lib\site-packages\sklearn\linear_model\logistic.py:433: FutureWarni
ng: Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence this warning.
  FutureWarning)
```

Out[28]:

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

In [36]:

```
prediction = logreg.predict(X_test)
```

In [ ]:

```

```

In [ ]:

```

```

In [37]:

```
from sklearn.metrics import classification_report
```

In [39]:

```
print(classification_report(y_test,prediction))
```

```
              precision    recall  f1-score   support

           0       0.86      0.96      0.90        89
           1       0.96      0.87      0.92       111

   micro avg       0.91      0.91      0.91       200
   macro avg       0.91      0.91      0.91       200
weighted avg       0.92      0.91      0.91       200
```

In [ ]:

```

```