Importing libraries

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

Imorting dataset

In [2]:
```
df=pd.read_csv("advertising_ef.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Gender	Country	Timestamp	Clicked on Ad
0	68.95	35.0	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	Female	Tunisia	27-03-2016 00:53	0
1	NaN	31.0	68441.85	193.77	Monitored national standardization	West Jodi	Male	Nauru	04-04-2016 01:39	0
2	69.47	26.0	59785.94	236.50	Organic bottom-line service-desk	Davidton	Female	San Marino	13-03-2016 20:35	0
3	74.15	29.0	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	Male	Italy	10-01-2016 02:31	0
4	68.37	35.0	73889.99	225.58	Robust logistical utilization	South Manuel	Female	Iceland	03-06-2016 03:36	0

In [4]:
```
df.tail()
```

Out[4]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Gender	Country	Timestamp	Clicked on Ad
1004	72.97	30.0	71384.57	208.58	Fundamental modular algorithm	Duffystad	Male	Lebanon	11-02-2016 21:49	1
1005	51.30	45.0	67782.17	134.42	Grass-roots cohesive monitoring	New Darlene	Male	Bosnia and Herzegovina	22-04-2016 02:07	1
1006	51.63	51.0	42415.72	120.37	Expanded intangible solution	South Jessica	Male	Mongolia	01-02-2016 17:24	1
1007	55.55	19.0	41920.79	187.95	Proactive bandwidth-monitored policy	West Steven	Female	Guatemala	24-03-2016 02:35	0
1008	45.01	26.0	29875.80	178.35	Virtual 5thgeneration emulation	Ronniemouth	Female	Brazil	03-06-2016 21:43	1

In [6]:
```
df.shape
```

Out[6]:
```
(1009, 10)
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1009 entries, 0 to 1008
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Daily Time Spent on Site  1005 non-null   float64
 1   Age                       998 non-null    float64
 2   Area Income               998 non-null    float64
 3   Daily Internet Usage      1005 non-null   float64
 4   Ad Topic Line             1009 non-null   object 
 5   City                      998 non-null    object 
 6   Gender                    1009 non-null   object 
 7   Country                   996 non-null    object 
 8   Timestamp                 1009 non-null   object 
 9   Clicked on Ad             1009 non-null   int64  
dtypes: float64(4), int64(1), object(5)
memory usage: 79.0+ KB
```

let's check whether data contains any null value or not

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
Daily Time Spent on Site     4
Age                         11
Area Income                 11
Daily Internet Usage         4
Ad Topic Line                0
City                        11
Gender                       0
Country                     13
Timestamp                    0
Clicked on Ad                0
dtype: int64
```

Managing missing or null values

In [10]:
```
df['Daily Time Spent on Site'].fillna(df['Daily Time Spent on Site'].mean(),inplace=True
```

In [11]:
```
df['Age'].fillna(df['Age'].mean(),inplace=True)
```

In [12]:
```
df['Daily Internet Usage'].fillna(df['Daily Internet Usage'].mean(),inplace=True)
```

In [13]:
```
df['Area Income'].fillna(df['Area Income'].mean(),inplace=True)
```

In [14]:
```
df.dropna(inplace=True)
```

- here we are able to remove/convert all the null values

In [16]:
```
df.isnull().sum()
```

Out[16]:
```
Daily Time Spent on Site    0
Age                         0
Area Income                 0
Daily Internet Usage        0
Ad Topic Line               0
City                        0
Gender                      0
Country                     0
Timestamp                   0
Clicked on Ad               0
dtype: int64
```

converting floats into int values

In [17]:
```
df["Daily Time Spent on Site"]=df["Daily Time Spent on Site"].astype('int')
```

In [19]:
```
df["Age"]=df["Age"].astype('int')
```

In [20]:
```
df["Area Income"]=df["Area Income"].astype('int')
```

In [21]:
```
df["Daily Internet Usage"]=df["Daily Internet Usage"].astype('int')
```

In [22]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 985 entries, 0 to 1008
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Daily Time Spent on Site  985 non-null    int32
 1   Age                       985 non-null    int32
 2   Area Income               985 non-null    int32
 3   Daily Internet Usage      985 non-null    int32
 4   Ad Topic Line             985 non-null    object
 5   City                      985 non-null    object
 6   Gender                    985 non-null    object
 7   Country                   985 non-null    object
 8   Timestamp                 985 non-null    object
 9   Clicked on Ad             985 non-null    int64
dtypes: int32(4), int64(1), object(5)
memory usage: 69.3+ KB
```

let's check whether data contains any duplicate values or not

In [24]:
```
df.duplicated()
```

Out[24]:
```
0       False
1       False
2       False
3       False
4       False
        ...
1004    False
1005    False
1006    False
1007    False
1008    False
Length: 985, dtype: bool
```

- Dataset doesn't contain any duplicated values

converting the "Timestamp" feature into year, month , day and hour

In [26]:
```
data=pd.DataFrame(df)
data["Timestamp"]=pd.date_range("1/1/2016 01:00:00",periods=985,freq="w")
print(data)
```

```
      Daily Time Spent on Site  Age  Area Income  Daily Internet Usage  \
0                           68   35        61833                   256
1                           65   31        68441                   193
2                           69   26        59785                   236
3                           74   29        54806                   245
4                           68   35        73889                   225
...                        ...  ...          ...                   ...
1004                        72   30        71384                   208
1005                        51   45        67782                   134
1006                        51   51        42415                   120
1007                        55   19        41920                   187
1008                        45   26        29875                   178

                            Ad Topic Line            City  Gender  \
0       Cloned 5thgeneration orchestration     Wrightburgh  Female
1       Monitored national standardization       West Jodi    Male
2         Organic bottom-line service-desk        Davidton  Female
3     Triple-buffered reciprocal time-frame  West Terrifurt    Male
4             Robust logistical utilization    South Manuel  Female
...                                     ...             ...     ...
1004          Fundamental modular algorithm       Duffystad    Male
1005        Grass-roots cohesive monitoring     New Darlene    Male
1006           Expanded intangible solution   South Jessica    Male
1007   Proactive bandwidth-monitored policy     West Steven  Female
1008        Virtual 5thgeneration emulation     Ronniemouth  Female

                     Country           Timestamp  Clicked on Ad
0                    Tunisia 2016-01-03 01:00:00              0
1                      Nauru 2016-01-10 01:00:00              0
2                  San Marino 2016-01-17 01:00:00              0
3                      Italy 2016-01-24 01:00:00              0
4                    Iceland 2016-01-31 01:00:00              0
...                      ...                 ...            ...
1004                 Lebanon 2034-10-15 01:00:00              1
1005  Bosnia and Herzegovina 2034-10-22 01:00:00              1
1006                Mongolia 2034-10-29 01:00:00              1
1007               Guatemala 2034-11-05 01:00:00              0
1008                  Brazil 2034-11-12 01:00:00              1

[985 rows x 10 columns]
```

In [27]:
```
data["year"]=data["Timestamp"].dt.year
```

In [28]:
```
data["month"]=data["Timestamp"].dt.month
```

In [29]:
```
data["day"]=data["Timestamp"].dt.day
```

In [30]:
```
data["hour"]=data["Timestamp"].dt.hour
```

In [31]:
```
del df["Timestamp"]
```

In [32]:
```
df.head()
```

Out[32]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Gender	Country	Clicked on Ad	year	month	day	hou
0	68	35	61833	256	Cloned 5thgeneration orchestration	Wrightburgh	Female	Tunisia	0	2016	1	3	1
1	65	31	68441	193	Monitored national standardization	West Jodi	Male	Nauru	0	2016	1	10	1
2	69	26	59785	236	Organic bottom-line service-desk	Davidton	Female	San Marino	0	2016	1	17	1
3	74	29	54806	245	Triple-buffered reciprocal time-frame	West Terrifurt	Male	Italy	0	2016	1	24	1
4	68	35	73889	225	Robust logistical utilization	South Manuel	Female	Iceland	0	2016	1	31	1

converting Gender column into numeric values

In [33]:
```
Gender_map={"Male":2,'Female':1}
```

In [34]:
```
df["Gender"].replace(Gender_map,inplace=True)
```

In [35]:
```
df.head()
```

Out[35]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Gender	Country	Clicked on Ad	year	month	day	hou
0	68	35	61833	256	Cloned 5thgeneration orchestration	Wrightburgh	1	Tunisia	0	2016	1	3	1
1	65	31	68441	193	Monitored national standardization	West Jodi	2	Nauru	0	2016	1	10	1
2	69	26	59785	236	Organic bottom-line service-desk	Davidton	1	San Marino	0	2016	1	17	1
3	74	29	54806	245	Triple-buffered reciprocal time-frame	West Terrifurt	2	Italy	0	2016	1	24	1
4	68	35	73889	225	Robust logistical utilization	South Manuel	1	Iceland	0	2016	1	31	1

separation of numeric & categorical Columns

In [36]:
```
numeric_columns=df.columns[df.dtypes!="object"]
categorical_columns=df.columns[df.dtypes=="object"]
```

```
...ns
```

Out[37]:
```
Index(['Daily Time Spent on Site', 'Age', 'Area Income',
       'Daily Internet Usage', 'Gender', 'Clicked on Ad', 'year', 'month',
       'day', 'hour'],
      dtype='object')
```

In [38]:
```
categorical_columns
```

Out[38]:
```
Index(['Ad Topic Line', 'City', 'Country'], dtype='object')
```

separation of numeric and categorical data

In [39]:
```
numeric_data=df.select_dtypes(include=[np.number])
categorical_data=df.select_dtypes(exclude=[np.number])
```

In [40]:
```
numeric_data
```

Out[40]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Gender	Clicked on Ad	year	month	day	hour
0	68	35	61833	256	1	0	2016	1	3	1
1	65	31	68441	193	2	0	2016	1	10	1
2	69	26	59785	236	1	0	2016	1	17	1
3	74	29	54806	245	2	0	2016	1	24	1
4	68	35	73889	225	1	0	2016	1	31	1
...	...	...	...	...	...	...	...	...	...	...
1004	72	30	71384	208	2	1	2034	10	15	1
1005	51	45	67782	134	2	1	2034	10	22	1
1006	51	51	42415	120	2	1	2034	10	29	1
1007	55	19	41920	187	1	0	2034	11	5	1
1008	45	26	29875	178	1	1	2034	11	12	1

985 rows × 10 columns

In [41]:
```
categorical_data
```

Out[41]:

	Ad Topic Line	City	Country
0	Cloned 5thgeneration orchestration	Wrightburgh	Tunisia
1	Monitored national standardization	West Jodi	Nauru
2	Organic bottom-line service-desk	Davidton	San Marino
3	Triple-buffered reciprocal time-frame	West Terrifurt	Italy
4	Robust logistical utilization	South Manuel	Iceland
...	...	...	...
1004	Fundamental modular algorithm	Duffystad	Lebanon
1005	Grass-roots cohesive monitoring	New Darlene	Bosnia and Herzegovina
1006	Expanded intangible solution	South Jessica	Mongolia
1007	Proactive bandwidth-monitored policy	West Steven	Guatemala
1008	Virtual 5thgeneration emulation	Ronniemouth	Brazil

985 rows × 3 columns

In [42]:
```
numeric_data.shape
```

Out[42]:
```
(985, 10)
```

In [43]:
```
categorical_data.shape
```

Out[43]:
```
(985, 3)
```

In [44]:
```
df.drop(["Ad Topic Line"],inplace=True,axis=1)
```

In [45]:
```
df.head()
```

Out[45]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	City	Gender	Country	Clicked on Ad	year	month	day	hour
0	68	35	61833	256	Wrightburgh	1	Tunisia	0	2016	1	3	1
1	65	31	68441	193	West Jodi	2	Nauru	0	2016	1	10	1
2	69	26	59785	236	Davidton	1	San Marino	0	2016	1	17	1
3	74	29	54806	245	West Terrifurt	2	Italy	0	2016	1	24	1
4	68	35	73889	225	South Manuel	1	Iceland	0	2016	1	31	1

statistical Analysis

In [46]:
```
# describe() method returns description of the data in the DataFrame (i.e. count, mean,
df.describe()
```

Out[46]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Gender	Clicked on Ad	year	month	
count	985.000000	985.000000	985.000000	985.000000	985.000000	985.000000	985.000000	985.000000	985
mean	64.446701	36.017259	54903.003046	179.164467	1.478173	0.506599	2024.938071	6.485279	15
std	15.857324	8.737052	13318.264216	43.868630	0.499777	0.500210	5.449996	3.438785	8
min	32.000000	19.000000	13996.000000	104.000000	1.000000	0.000000	2016.000000	1.000000	1
25%	51.000000	29.000000	46974.000000	138.000000	1.000000	0.000000	2020.000000	4.000000	8
50%	68.000000	35.000000	56735.000000	181.000000	1.000000	1.000000	2025.000000	7.000000	16
75%	78.000000	42.000000	65227.000000	218.000000	2.000000	1.000000	2030.000000	9.000000	23
max	91.000000	61.000000	79484.000000	269.000000	2.000000	1.000000	2034.000000	12.000000	31

In [47]:
```
numeric_data
```

Out[47]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Gender	Clicked on Ad	year	month	day	hour
0	68	35	61833	256	1	0	2016	1	3	1
1	65	31	68441	193	2	0	2016	1	10	1
2	69	26	59785	236	1	0	2016	1	17	1
3	74	29	54806	245	2	0	2016	1	24	1
4	68	35	73889	225	1	0	2016	1	31	1
...	...	...	...	...	...	...	...	...	...	...
1004	72	30	71384	208	2	1	2034	10	15	1
1005	51	45	67782	134	2	1	2034	10	22	1
1006	51	51	42415	120	2	1	2034	10	29	1
1007	55	19	41920	187	1	0	2034	11	5	1
1008	45	26	29875	178	1	1	2034	11	12	1

985 rows × 10 columns

mean, median ,mode

In [48]:
```
np.mean(numeric_data)
```

```
C:\Users\SHREE\anaconda3\lib\site-packages\numpy\core\fromnumeric.py:3430: FutureWarnin
g: In a future version, DataFrame.mean(axis=None) will return a scalar mean over the ent
ire DataFrame. To retain the old behavior, use 'frame.mean(axis=0)' or just 'frame.mean
()'
  return mean(axis=axis, dtype=dtype, out=out, **kwargs)
```

Out[48]:
```
Daily Time Spent on Site       64.446701
Age                            36.017259
Area Income                 54903.003046
Daily Internet Usage          179.164467
Gender                          1.478173
Clicked on Ad                   0.506599
year                         2024.938071
month                           6.485279
day                            15.727919
hour                            1.000000
dtype: float64
```

In [49]:
```
np.median(numeric_data)
```

Out[49]:
```
27.0
```

In [50]:
```
import statistics

statistics.mode(numeric_data)
```

Out[50]:
```
'Daily Time Spent on Site'
```

dispersion

In [51]:
```
np.percentile(numeric_data,[25])
```

Out[51]:
```
array([1.])
```

In [52]:
```
np.percentile(numeric_data,[25,50,75,100])
```

Out[52]:
```
array([1.0000e+00, 2.7000e+01, 1.8100e+02, 7.9484e+04])
```

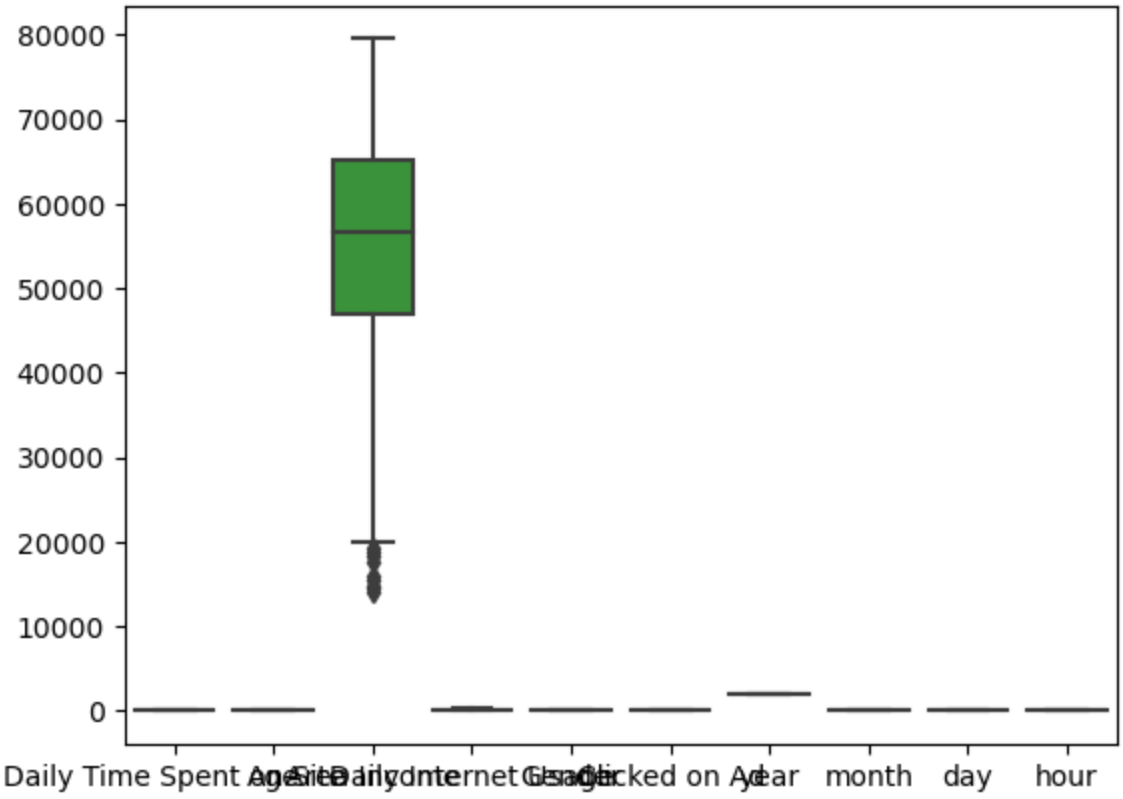
q1,q2,q3,q4

iqr=q3-q1

BOXPLOT visualization to check whether any outliers present or not

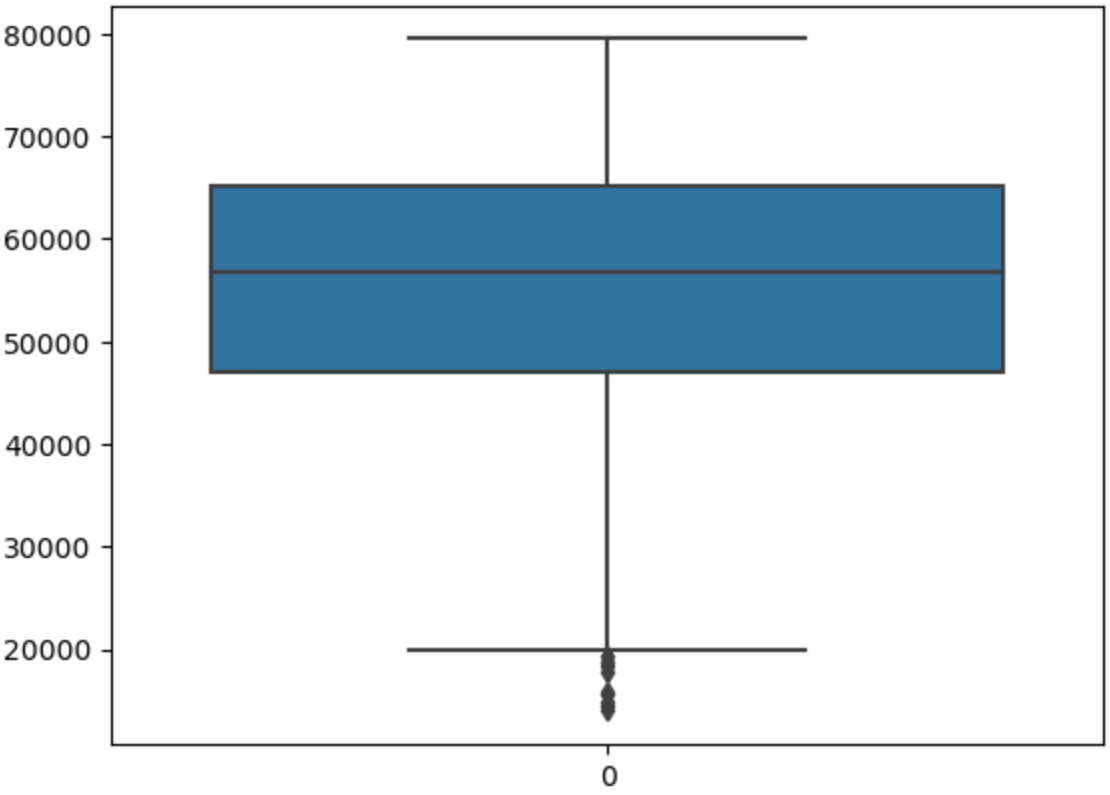
In [53]:
```
import seaborn as sns
sns.boxplot(numeric_data)
```

Out[53]:
```
<Axes: >
```



In [54]:
```
import seaborn as sns
sns.boxplot(numeric_data['Area Income'])
```

Out[54]:
```
<Axes: >
```



position of the outliers

In [55]:
```
print(np.where(numeric_data['Area Income']<20000))
```

```
(array([124, 374, 395, 495, 622, 647, 674, 750, 760, 936], dtype=int64),)
```

removal of outliers

In [56]:
```
# Create the dataframe
column_name = df['Area Income']
df_diabetes = pd.DataFrame(df)
df.head()
print("Old Shape: ", df.shape)

''' Detection '''
# IQR
# Calculate the upper and lower limits
Q1 = df['Area Income'].quantile(0.25)
Q3 = df['Area Income'].quantile(0.75)
IQR = Q3 - Q1
lower = Q1 - 1.5*IQR
upper = Q3 + 1.5*IQR

# Create arrays of Boolean values indicating the outlier rows
upper_array = np.where(df['Area Income']>=upper)[0]
lower_array = np.where(df['Area Income']<=lower)[0]

# Removing the outliers
df.drop(index=upper_array, inplace=True)
df.drop(index=lower_array, inplace=True)

# Print the new shape of the DataFrame
print("New Shape: ", df.shape)
```

```
Old Shape:  (985, 12)
New Shape:  (976, 12)
```

covariance and corelation

In [57]:
```
df.drop(["hour"],axis=1,inplace=True)
```

In [58]:
```
df.cov()
```

```
C:\Users\SHREE\AppData\Local\Temp\ipykernel_6296\1545644723.py:1: FutureWarning: The def
ault value of numeric_only in DataFrame.cov is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only to sile
nce this warning.
  df.cov()
```

Out[58]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Gender	Clicked on Ad	year	
Daily Time Spent on Site	252.881168	-45.603760	6.519108e+04	355.155241	-0.235288	-5.896752	-3.532141	
Age	-45.603760	75.895179	-2.016635e+04	-140.187417	-0.083808	2.109641	0.180620	
Area Income	65191.083153	-20166.349424	1.780955e+08	192270.254351	32.382547	-3134.447020	-4459.178167	-110
Daily Internet Usage	355.155241	-140.187417	1.922703e+05	1921.844446	0.452867	-17.244991	-1.801001	
Gender	-0.235288	-0.083808	3.238255e+01	0.452867	0.249836	-0.007545	0.186587	
Clicked on Ad	-5.896752	2.109641	-3.134447e+03	-17.244991	-0.007545	0.250205	0.067746	
year	-3.532141	0.180620	-4.459178e+03	-1.801001	0.186587	0.067746	29.789667	
month	-3.131184	1.244088	-1.109083e+03	-6.721592	0.102072	0.105634	-0.356329	1
day	-0.771198	-2.515998	-1.182366e+03	11.768424	-0.234435	-0.092358	-0.337907	

In [60]:
```
df.corr()
```

```
C:\Users\SHREE\AppData\Local\Temp\ipykernel_6296\1134722465.py:1: FutureWarning: The def
ault value of numeric_only in DataFrame.corr is deprecated. In a future version, it will
default to False. Select only valid columns or specify the value of numeric_only to sile
nce this warning.
  df.corr()
```

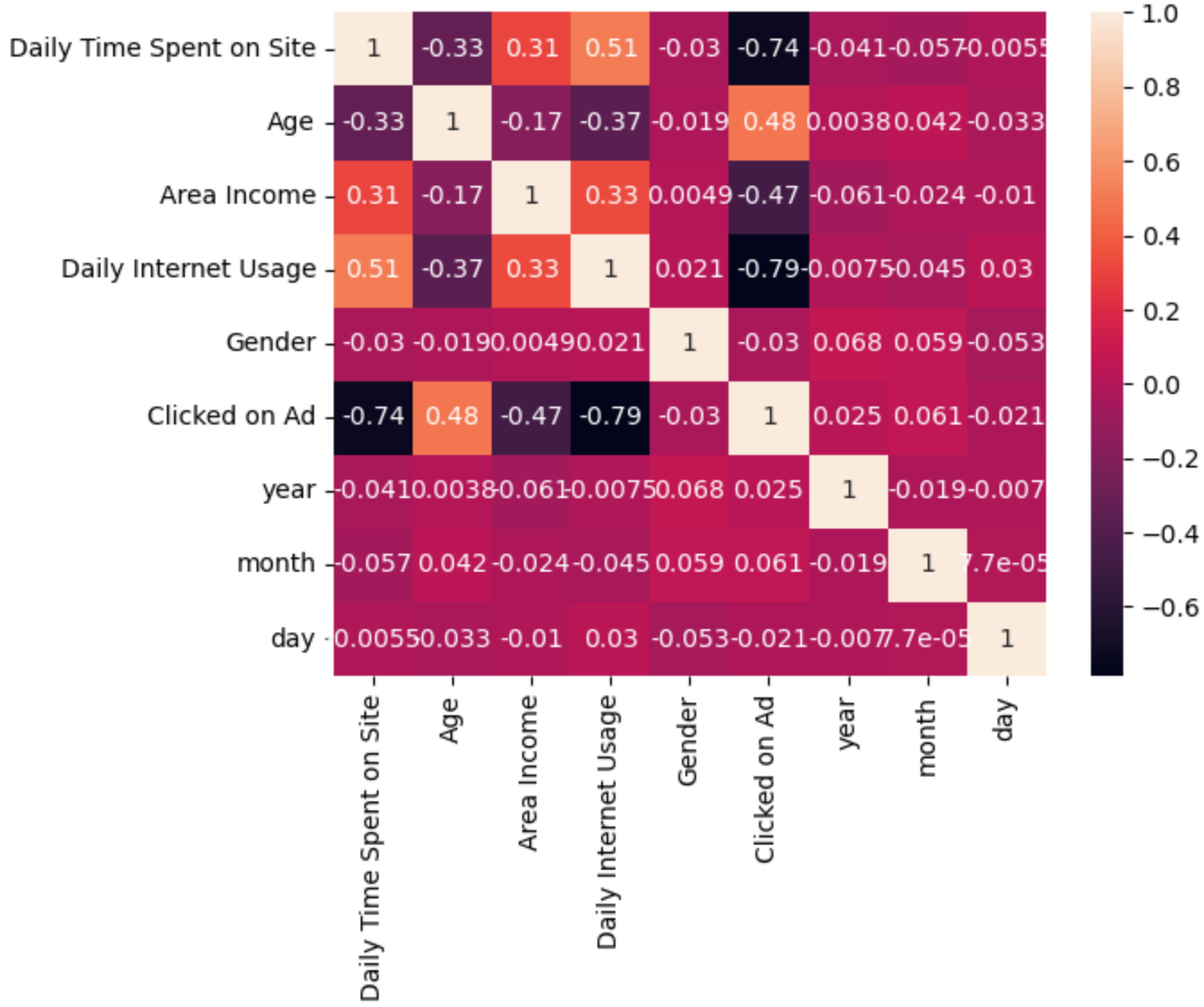
Out[60]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Gender	Clicked on Ad	year	month	day
Daily Time Spent on Site	1.000000	-0.329181	0.307187	0.509449	-0.029602	-0.741322	-0.040696	-0.057308	-0.005509
Age	-0.329181	1.000000	-0.173458	-0.367065	-0.019247	0.484120	0.003799	0.041563	-0.032808
Area Income	0.307187	-0.173458	1.000000	0.328645	0.004855	-0.469555	-0.061220	-0.024188	-0.010065
Daily Internet Usage	0.509449	-0.367065	0.328645	1.000000	0.020667	-0.786422	-0.007527	-0.044625	0.030496
Gender	-0.029602	-0.019247	0.004855	0.020667	1.000000	-0.030178	0.068394	0.059435	-0.053281
Clicked on Ad	-0.741322	0.484120	-0.469555	-0.786422	-0.030178	1.000000	0.024814	0.061464	-0.020975
year	-0.040696	0.003799	-0.061220	-0.007527	0.068394	0.024814	1.000000	-0.019001	-0.007033
month	-0.057308	0.041563	-0.024188	-0.044625	0.059435	0.061464	-0.019001	1.000000	0.000077
day	-0.005509	-0.032808	-0.010065	0.030496	-0.053281	-0.020975	-0.007033	0.000077	1.000000

In [61]:
```
### heatmap
sns.heatmap(df.corr(),annot=True)
```

```
C:\Users\SHREE\AppData\Local\Temp\ipykernel_6296\3374984919.py:2: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
  sns.heatmap(df.corr(),annot=True)
```

Out[61]: <Axes: >



Graph

In [63]:
```
df.head(3)
```

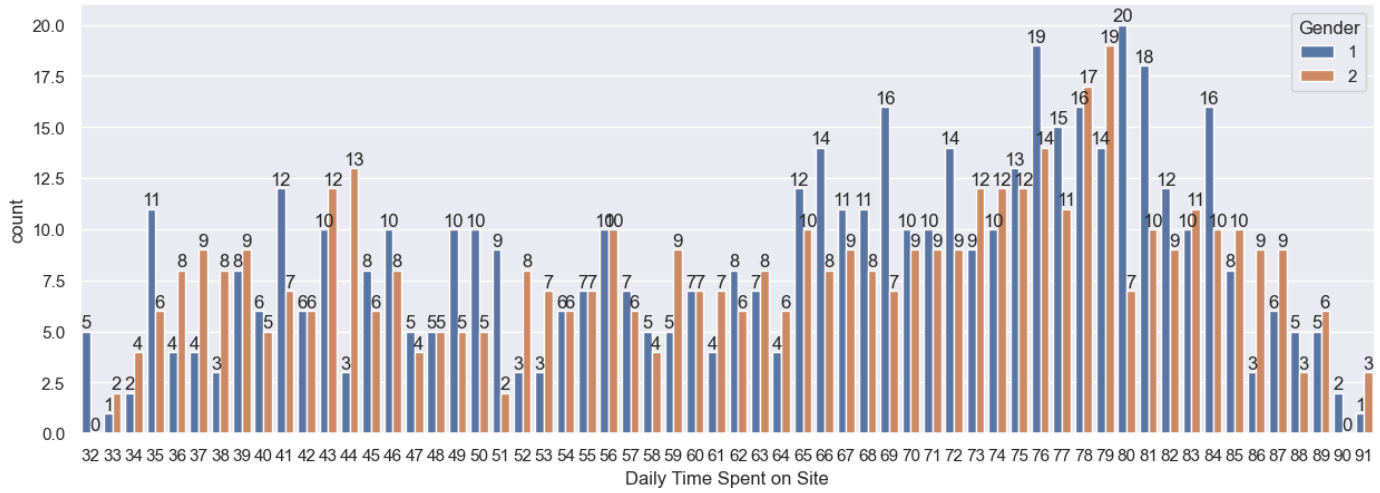
Out[63]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	City	Gender	Country	Clicked on Ad	year	month	day
0	68	35	61833	256	Wrightburgh	1	Tunisia	0	2016	1	3
1	65	31	68441	193	West Jodi	2	Nauru	0	2016	1	10
2	69	26	59785	236	Davidton	1	San Marino	0	2016	1	17

Daily Time Spent on Site

In [149…
```
ax = sns.countplot(data = df, x = 'Daily Time Spent on Site', hue = 'Gender')
```

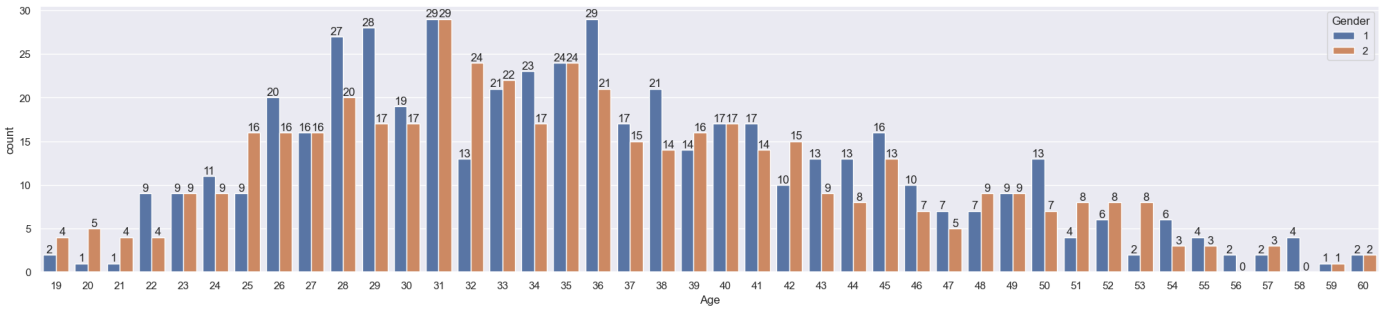
```
for bars in ax.containers:
    ax.bar_label(bars)
    sns.set(rc={'figure.figsize':(25,5)})
```



This perticular plot shows that most of the females are daily spend time on site.

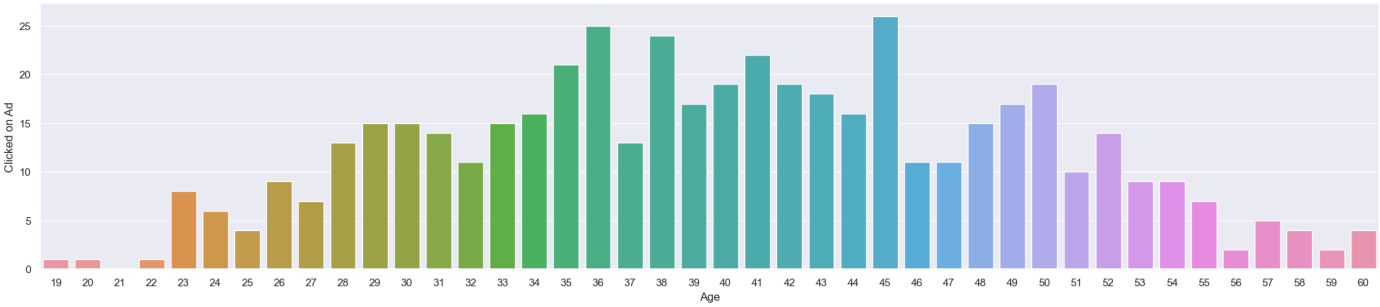
Age

In [77]:
```
ax = sns.countplot(data = df, x = 'Age', hue = 'Gender')

for bars in ax.containers:
    ax.bar_label(bars)
    sns.set(rc={'figure.figsize':(25,5)})
```



This plot shows that age between 26 to 38 mostly females are clicked on ad.

In [76]:
```
clicked_on_ad= df.groupby(['Age'], as_index=False)['Clicked on Ad'].sum().sort_values(by

sns.barplot(x = 'Age',y= 'Clicked on Ad' ,data = clicked_on_ad
sns.set(rc={'figure.figsize':(25,5)})
```

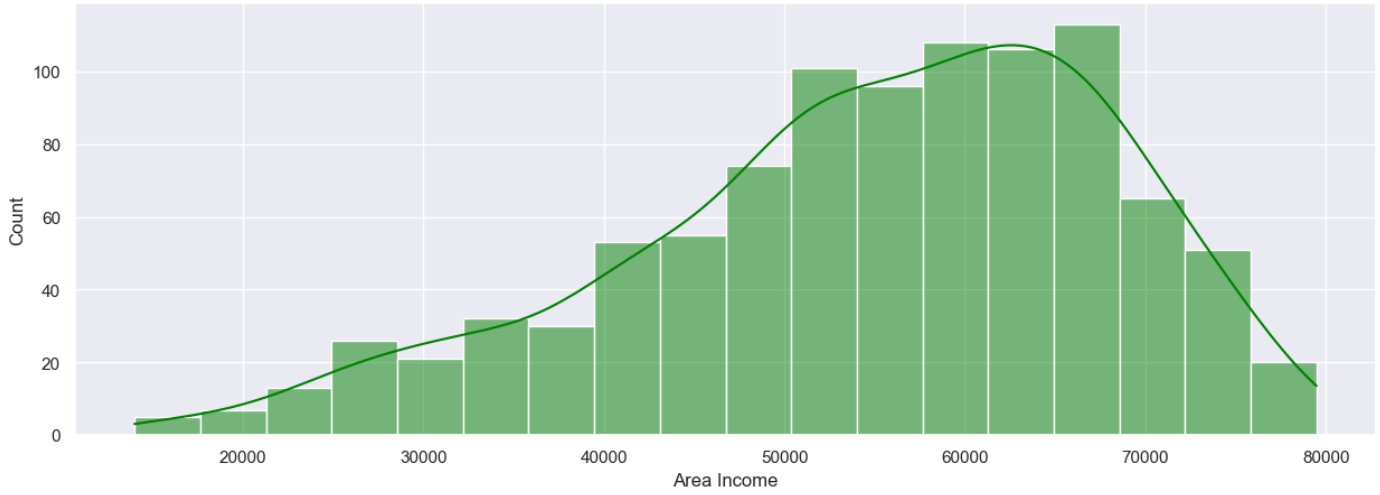


This plot explains that at age 36,38 and 45 people clicked on ad mostly.

Area Income

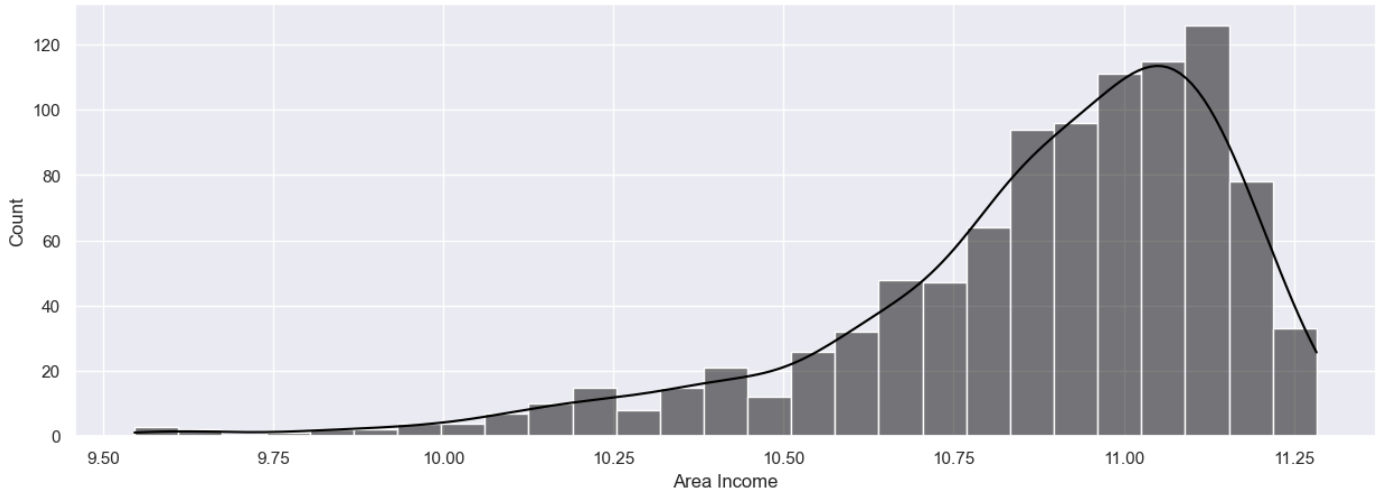
In [147…
```
sns.histplot(df['Area Income'],kde=True,color='green')
```

Out[147]:
```
<Axes: xlabel='Area Income', ylabel='Count'>
```



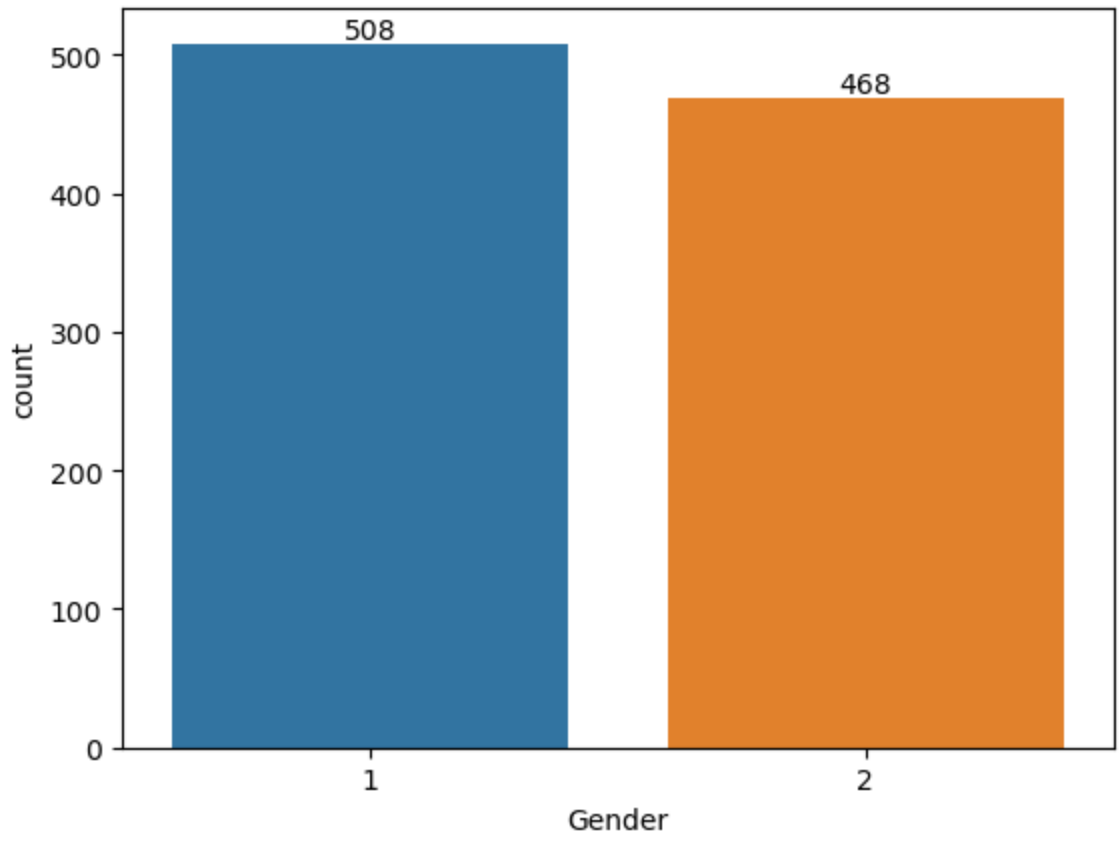
In [148…
```
sns.histplot(np.log(df['Area Income']),kde=True,color='black')
```

Out[148]:
```
<Axes: xlabel='Area Income', ylabel='Count'>
```



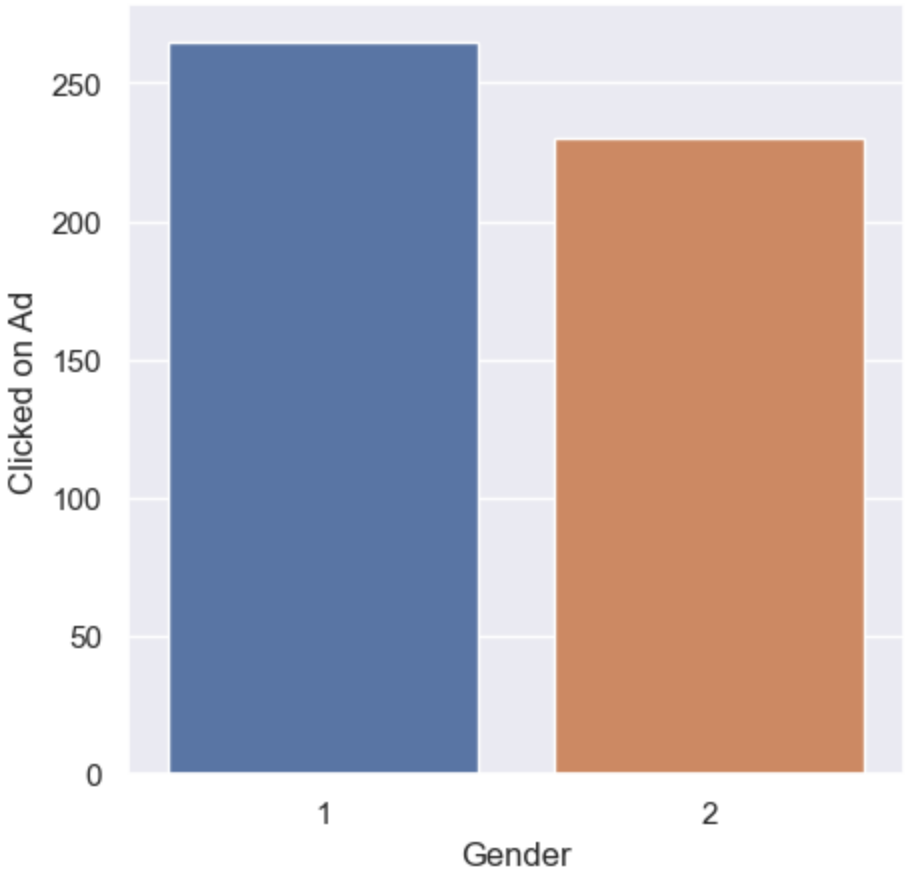
Gender

In [62]:
```
# plotting a bar chart for Gender and it's count

ax = sns.countplot(x = 'Gender',data = df)

for bars in ax.containers:
    ax.bar_label(bars)
```



In [146…

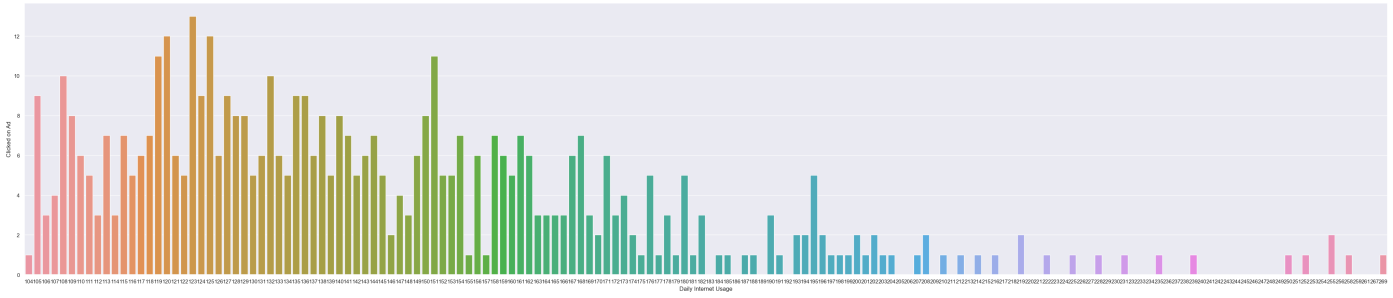
```
clicked_on_ad= df.groupby(['Gender'], as_index=False)['Clicked on Ad'].sum().sort_values

sns.barplot(x = 'Gender',y= 'Clicked on Ad' ,data = clicked_on_ad)
sns.set(rc={'figure.figsize':(15,5)})
```



From above graphs we can see that most of the Females are clicked on add

Daily Internet Usage

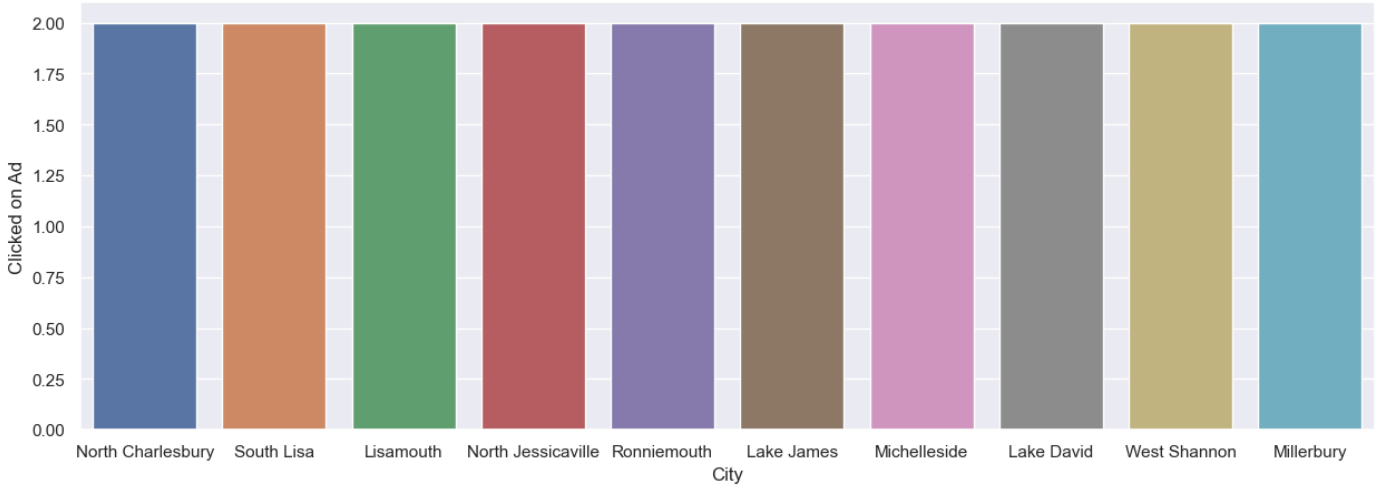
In [89]:
```
sales_gen = df.groupby(['Daily Internet Usage'], as_index=False)['Clicked on Ad'].sum().

sns.barplot(x = 'Daily Internet Usage',y= 'Clicked on Ad' ,data = sales_gen)
sns.set(rc={'figure.figsize':(50,4)})
```



City

In [92]:
```
sales_state = df.groupby(['City'], as_index=False)['Clicked on Ad'].sum().sort_values(by

sns.set(rc={'figure.figsize':(15,5)})
sns.barplot(data = sales_state, x = 'City',y= 'Clicked on Ad')
```

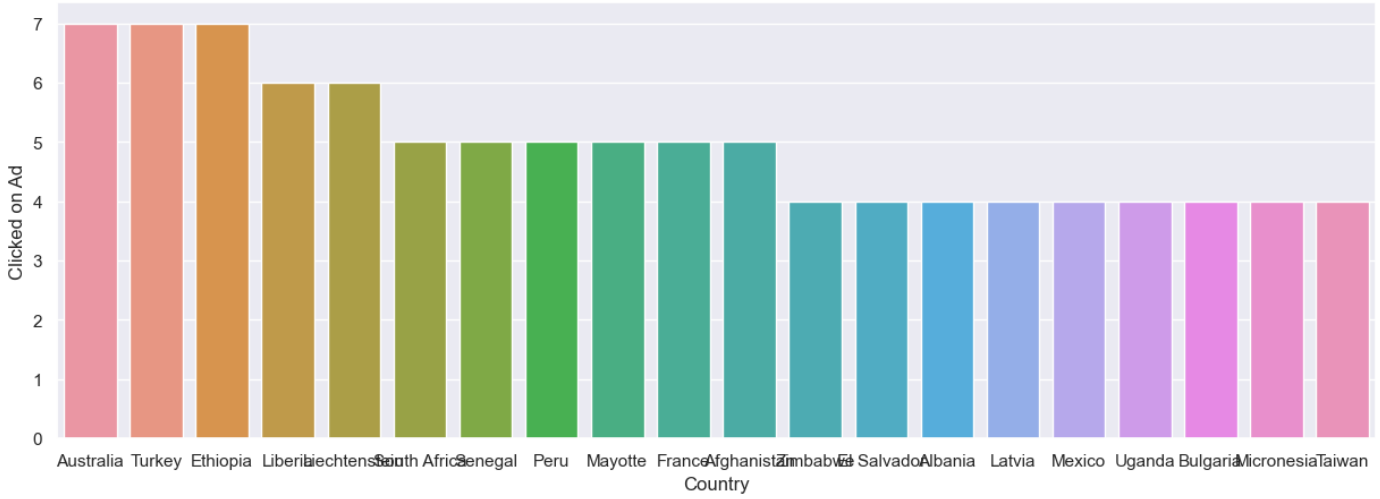
Out[92]:
```
<Axes: xlabel='City', ylabel='Clicked on Ad'>
```



Country

In [94]:
```
sales_state = df.groupby(['Country'], as_index=False)['Clicked on Ad'].sum().sort_values

sns.set(rc={'figure.figsize':(15,5)})
sns.barplot(data = sales_state, x = 'Country',y= 'Clicked on Ad')
```

Out[94]:
```
<Axes: xlabel='Country', ylabel='Clicked on Ad'>
```

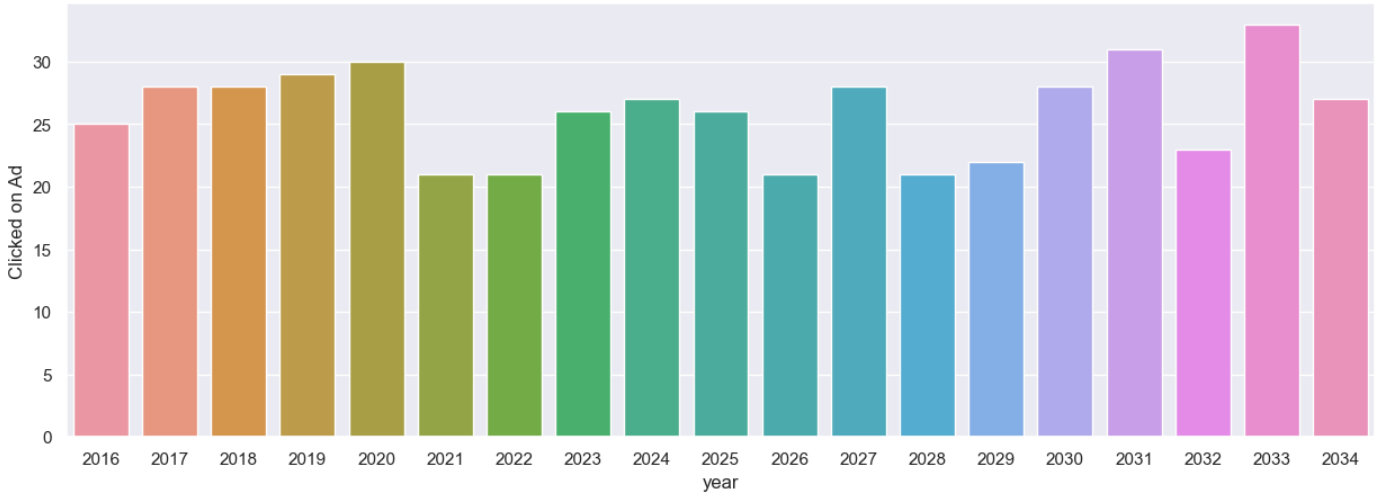


This perticular plot shows that Australia then Turkey and Ethiopia are the countries in which most of the people clicked on ad.

year

In [96]:
```
sales_state = df.groupby(['year'], as_index=False)['Clicked on Ad'].sum().sort_values(by

sns.set(rc={'figure.figsize':(15,5)})
sns.barplot(data = sales_state, x = 'year',y= 'Clicked on Ad')
```

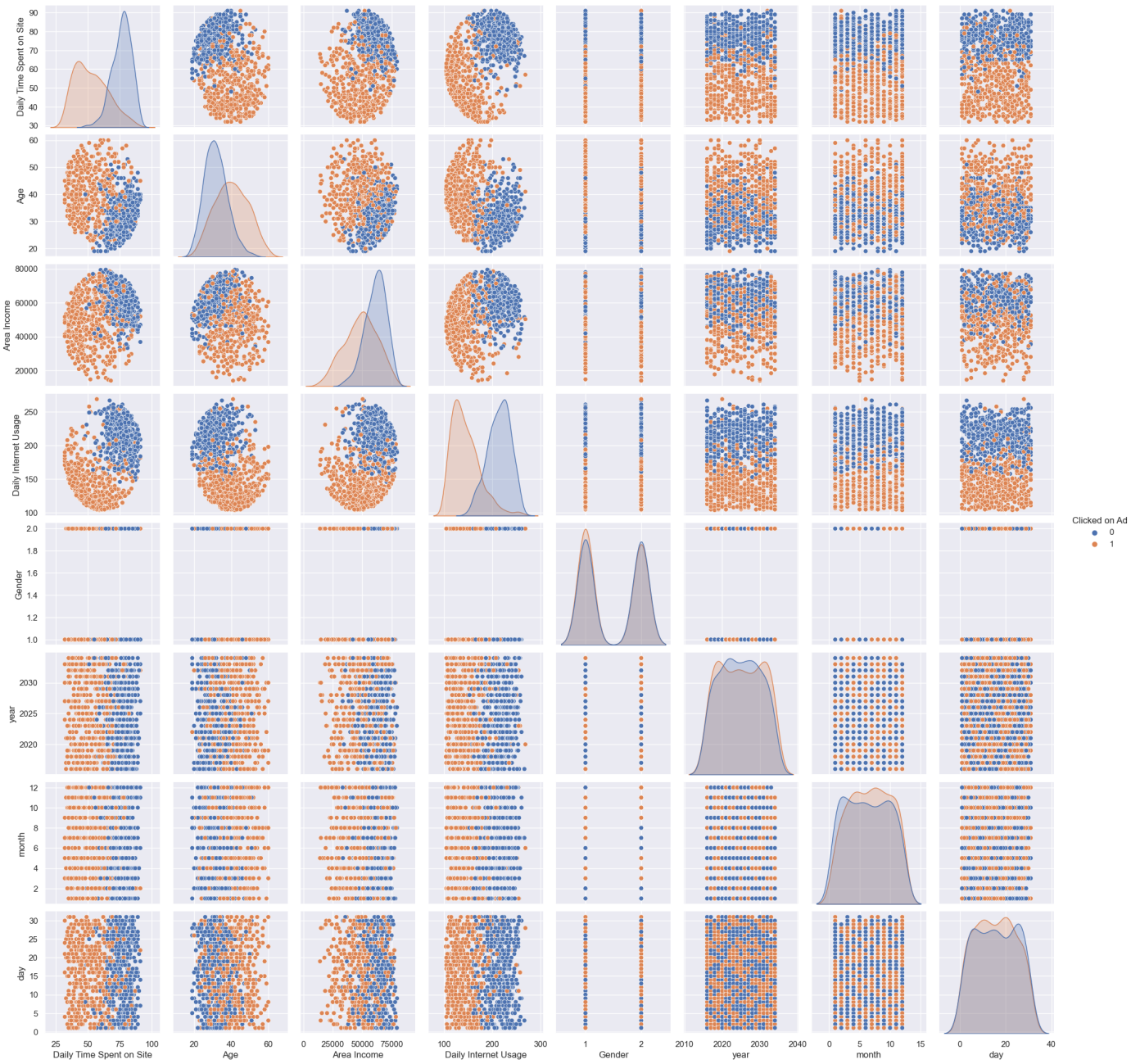
Out[96]:
```
<Axes: xlabel='year', ylabel='Clicked on Ad'>
```



This plot shows that in the year 2033 people mostly clicked on add

In [97]:
```
# QUICK EDA
import seaborn as sns
sns.pairplot(df,hue="Clicked on Ad")
```

Out[97]:
```
<seaborn.axisgrid.PairGrid at 0x29e78897ca0>
```



Conclusion= here we can conclude that in the year of 2033 in Australia,Turkey & Ethiopia most of the female customers are clicked on ad.