In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

In [2]:

```
# You have to include the full link to the csv file containing your dataset
employee_df = pd.read_csv('D:/Data Science for Business Package/1. Human Resources D
employee_df
```

Out[2]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Educatior
0	41	Yes	Travel_Rarely	1102	Sales	1	2
1	49	No	Travel_Frequently	279	Research & Development	8	1
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2
3	33	No	Travel_Frequently	1392	Research & Development	3	4
4	27	No	Travel_Rarely	591	Research & Development	2	1
...	...	...	...	...	...	...	..
1465	36	No	Travel_Frequently	884	Research & Development	23	2
1466	39	No	Travel_Rarely	613	Research & Development	6	1
1467	27	No	Travel_Rarely	155	Research & Development	4	3
1468	49	No	Travel_Frequently	1023	Sales	2	3
1469	34	No	Travel_Rarely	628	Research & Development	8	3

1470 rows × 35 columns

In [3]:

```
print(employee_df.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype 
---  ------                    --------------  ----- 
 0   Age                       1470 non-null   int64 
 1   Attrition                 1470 non-null   object
 2   BusinessTravel            1470 non-null   object
 3   DailyRate                 1470 non-null   int64 
 4   Department                1470 non-null   object
 5   DistanceFromHome          1470 non-null   int64 
 6   Education                 1470 non-null   int64 
 7   EducationField            1470 non-null   object
 8   EmployeeCount             1470 non-null   int64 
 9   EmployeeNumber            1470 non-null   int64 
 10  EnvironmentSatisfaction   1470 non-null   int64 
 11  Gender                    1470 non-null   object
 12  HourlyRate                1470 non-null   int64 
 13  JobInvolvement            1470 non-null   int64 
 14  JobLevel                  1470 non-null   int64 
 15  JobRole                   1470 non-null   object
 16  JobSatisfaction           1470 non-null   int64 
 17  MaritalStatus             1470 non-null   object
 18  MonthlyIncome             1470 non-null   int64 
 19  MonthlyRate               1470 non-null   int64 
 20  NumCompaniesWorked        1470 non-null   int64 
 21  Over18                    1470 non-null   object
 22  OverTime                  1470 non-null   object
 23  PercentSalaryHike         1470 non-null   int64 
 24  PerformanceRating         1470 non-null   int64 
 25  RelationshipSatisfaction  1470 non-null   int64 
 26  StandardHours             1470 non-null   int64 
 27  StockOptionLevel          1470 non-null   int64 
 28  TotalWorkingYears         1470 non-null   int64 
 29  TrainingTimesLastYear     1470 non-null   int64 
 30  WorkLifeBalance           1470 non-null   int64 
 31  YearsAtCompany            1470 non-null   int64 
 32  YearsInCurrentRole        1470 non-null   int64 
 33  YearsSinceLastPromotion   1470 non-null   int64 
 34  YearsWithCurrManager      1470 non-null   int64 
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
None
```

In [4]:

```
print("Attrition columns record values",employee_df['Attrition'].unique())
print("Overtime columns record values",employee_df['OverTime'].unique())
print("Over18 columns record values",employee_df['Over18'].unique())

#categorical data into numerical transformation manually
employee_df['Attrition'] = employee_df['Attrition'].apply(lambda x: 1 if x == 'Yes'
employee_df['OverTime'] = employee_df['OverTime'].apply(lambda x: 1 if x == 'Yes' el
employee_df['Over18'] = employee_df['Over18'].apply(lambda x: 1 if x == 'Y' else 0)

```

```
Attrition columns record values ['Yes' 'No']
Overtime columns record values ['Yes' 'No']
Over18 columns record values ['Y']
```

In [5]:

```
#checking for null values
employee_df.isnull().sum()

```

Out[5]:

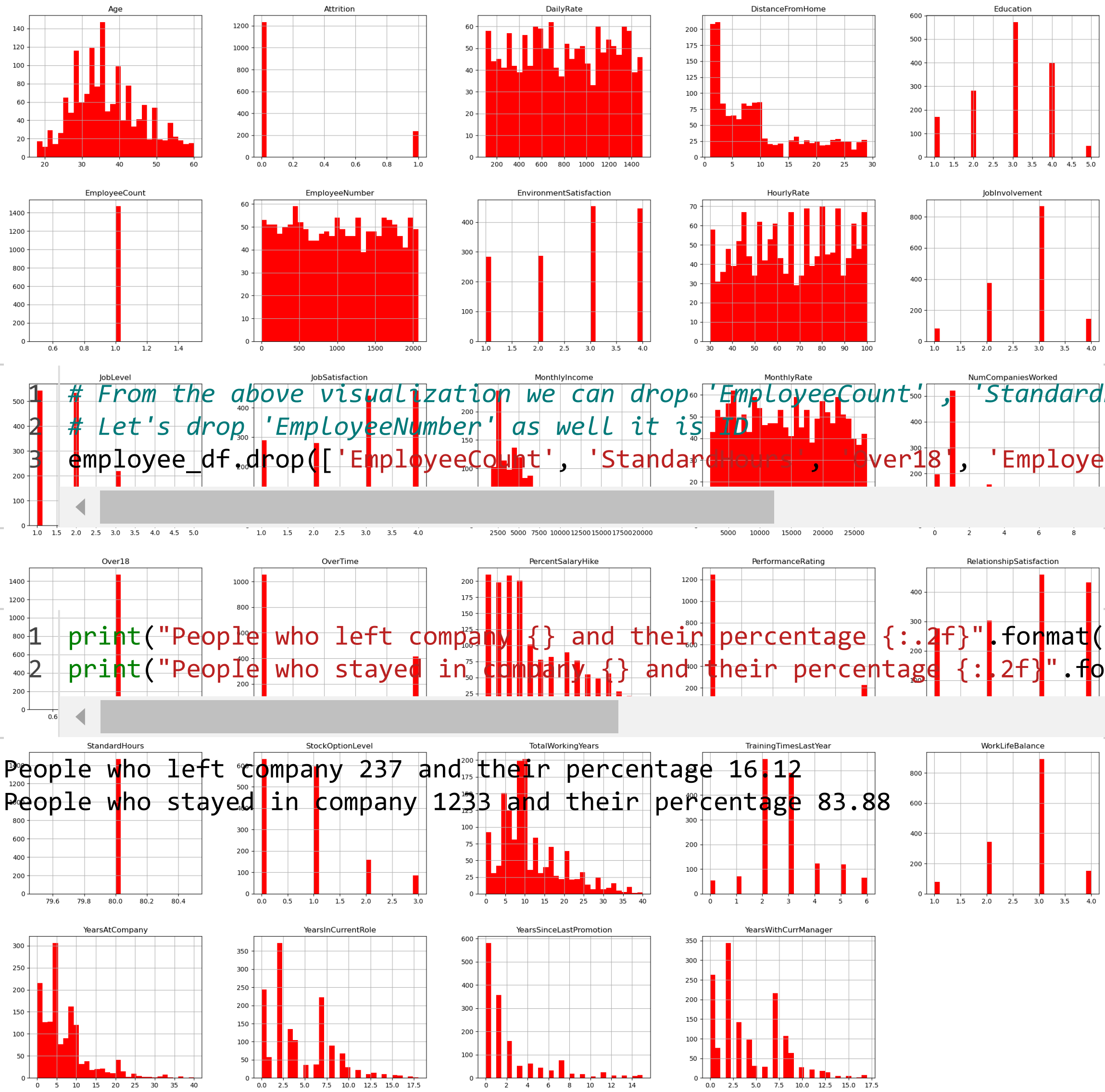
```
Age                         0
Attrition                   0
BusinessTravel              0
DailyRate                   0
Department                  0
DistanceFromHome            0
Education                   0
EducationField              0
EmployeeCount               0
EmployeeNumber              0
EnvironmentSatisfaction     0
Gender                      0
HourlyRate                  0
JobInvolvement              0
JobLevel                    0
JobRole                     0
JobSatisfaction             0
MaritalStatus               0
MonthlyIncome               0
MonthlyRate                 0
NumCompaniesWorked          0
Over18                      0
OverTime                    0
PercentSalaryHike           0
PerformanceRating           0
RelationshipSatisfaction    0
StandardHours               0
StockOptionLevel            0
TotalWorkingYears           0
TrainingTimesLastYear       0
WorkLifeBalance             0
YearsAtCompany              0
YearsInCurrentRole          0
YearsSinceLastPromotion     0
YearsWithCurrManager        0
dtype: int64
```

In [6]:

```
employee_df.hist(bins = 30,figsize = (30,30), color = 'r')
```

Out[6]:

```
array([[<AxesSubplot: title={'center': 'Age'}>,
        <AxesSubplot: title={'center': 'Attrition'}>,
        <AxesSubplot: title={'center': 'DailyRate'}>,
        <AxesSubplot: title={'center': 'DistanceFromHome'}>,
        <AxesSubplot: title={'center': 'Education'}>],
       [<AxesSubplot: title={'center': 'EmployeeCount'}>,
        <AxesSubplot: title={'center': 'EmployeeNumber'}>,
        <AxesSubplot: title={'center': 'EnvironmentSatisfaction'}>,
        <AxesSubplot: title={'center': 'HourlyRate'}>,
        <AxesSubplot: title={'center': 'JobInvolvement'}>],
       [<AxesSubplot: title={'center': 'JobLevel'}>,
        <AxesSubplot: title={'center': 'JobSatisfaction'}>,
        <AxesSubplot: title={'center': 'MonthlyIncome'}>,
        <AxesSubplot: title={'center': 'MonthlyRate'}>,
        <AxesSubplot: title={'center': 'NumCompaniesWorked'}>],
       [<AxesSubplot: title={'center': 'Over18'}>,
        <AxesSubplot: title={'center': 'OverTime'}>,
        <AxesSubplot: title={'center': 'PercentSalaryHike'}>,
        <AxesSubplot: title={'center': 'PerformanceRating'}>,
        <AxesSubplot: title={'center': 'RelationshipSatisfaction'}>],
       [<AxesSubplot: title={'center': 'StandardHours'}>,
        <AxesSubplot: title={'center': 'StockOptionLevel'}>,
        <AxesSubplot: title={'center': 'TotalWorkingYears'}>,
        <AxesSubplot: title={'center': 'TrainingTimesLastYear'}>,
        <AxesSubplot: title={'center': 'WorkLifeBalance'}>],
       [<AxesSubplot: title={'center': 'YearsAtCompany'}>,
        <AxesSubplot: title={'center': 'YearsInCurrentRole'}>,
        <AxesSubplot: title={'center': 'YearsSinceLastPromotion'}>,
        <AxesSubplot: title={'center': 'YearsWithCurrManager'}>,
        <AxesSubplot: >]], dtype=object)
```




```
# From the above visualization we can drop 'EmployeeCount' , 'Standardhours' and 'Ov
# Let's drop 'EmployeeNumber' as well it is ID
employee_df.drop(['EmployeeCount', 'StandardHours', 'Over18', 'EmployeeNumber'], axi
```

```
print("People who left company {} and their percentage {:.2f}".format(len(employee_c
print("People who stayed in company {} and their percentage {:.2f}".format(len(emplo
```

```
People who left company 237 and their percentage 16.12
People who stayed in company 1233 and their percentage 83.88
```

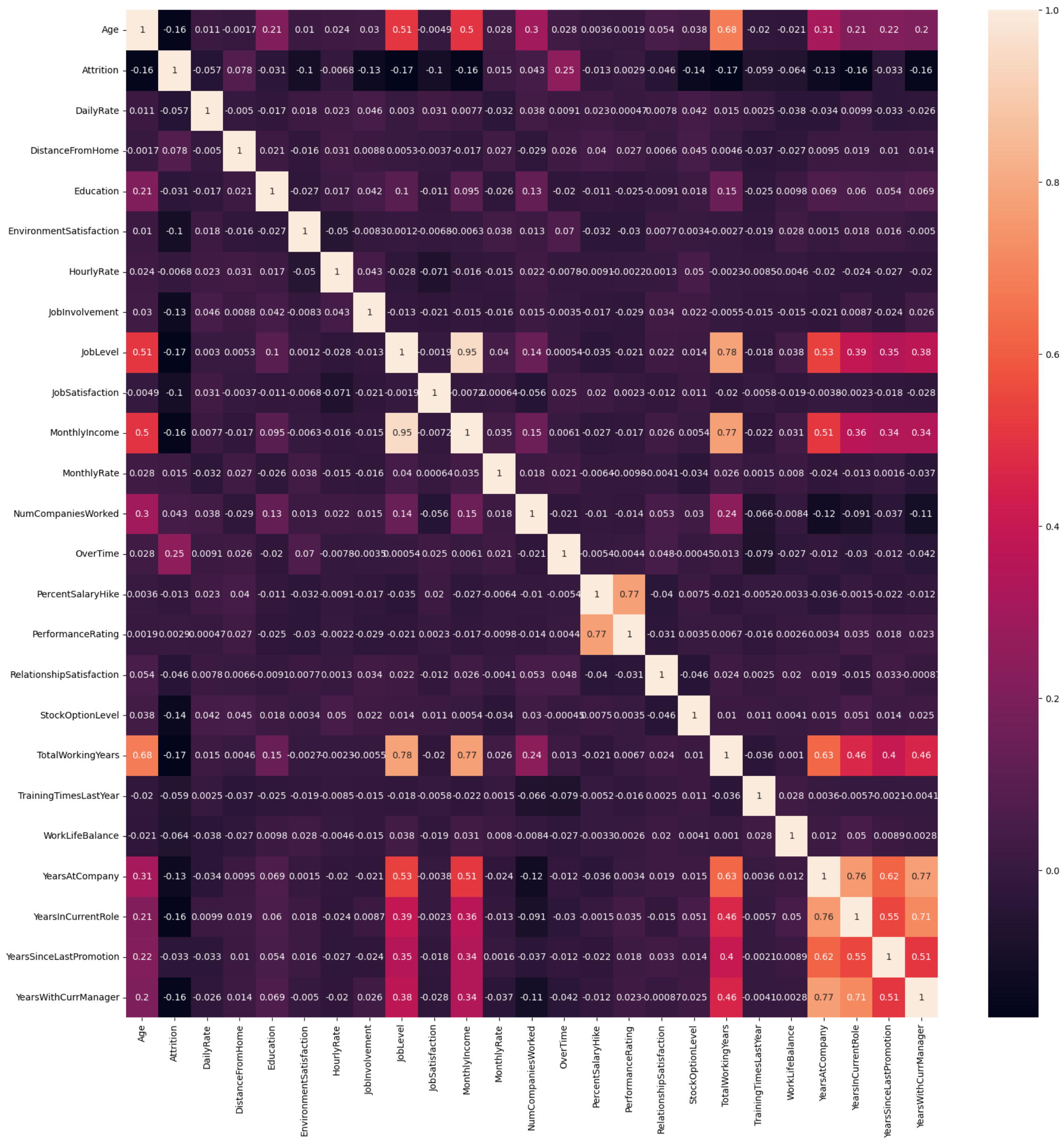
In [9]:

```
#correlation plot

correlations = employee_df.corr()
fig, ax = plt.subplots(figsize = (20, 20))
sns.heatmap(correlations, annot = True)
```

```
C:\Users\Pradeep\AppData\Local\Temp\ipykernel_22596\3969665670.py:3: Futur
eWarning: The default value of numeric_only in DataFrame.corr is deprecate
d. In a future version, it will default to False. Select only valid column
s or specify the value of numeric_only to silence this warning.
  correlations = employee_df.corr()
```

Out[9]:

```
<AxesSubplot: >
```

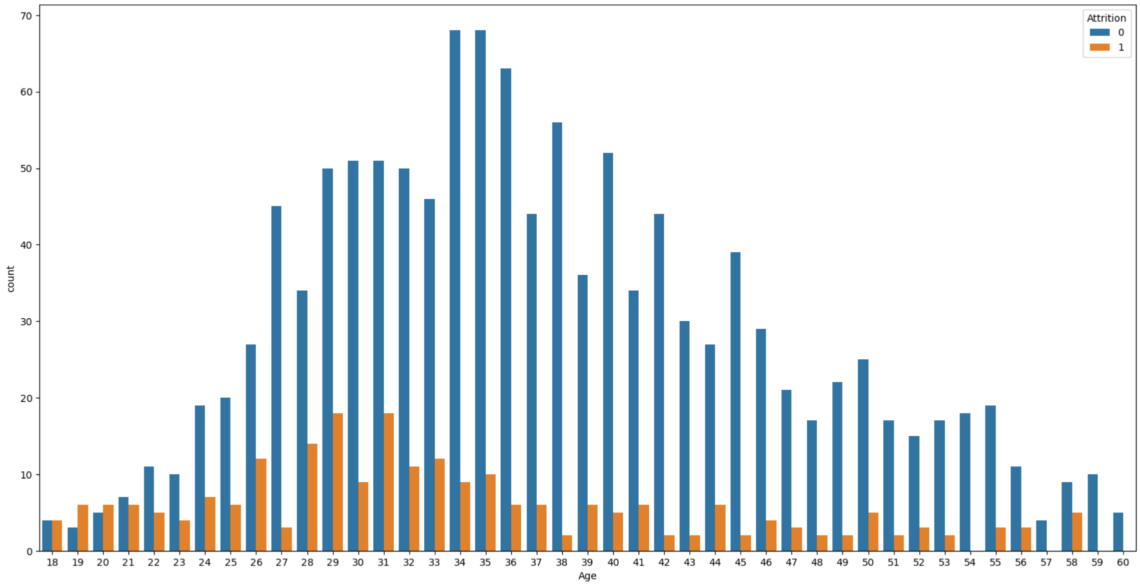


In [10]:

```
#visualizing age of people who stayed and left the company
plt.figure(figsize=[20,10])
sns.countplot(x = 'Age', hue = 'Attrition', data = employee_df)
```

Out[10]:

<AxesSubplot: xlabel='Age', ylabel='count'>



In [11]:

```
employee_df.describe()
```

Out[11]:

	Age	Attrition	DailyRate	DistanceFromHome	Education	Environment
count	1470.000000	1470.000000	1470.000000	1470.000000	1470.000000	
mean	36.923810	0.161224	802.485714	9.192517	2.912925	
std	9.135373	0.367863	403.509100	8.106864	1.024165	
min	18.000000	0.000000	102.000000	1.000000	1.000000	
25%	30.000000	0.000000	465.000000	2.000000	2.000000	
50%	36.000000	0.000000	802.000000	7.000000	3.000000	
75%	43.000000	0.000000	1157.000000	14.000000	4.000000	
max	60.000000	1.000000	1499.000000	29.000000	5.000000	

8 rows × 25 columns

In [12]:

```
#encoding for categorical column
employee_df = pd.get_dummies(employee_df,columns=['Department','BusinessTravel','Edu
employee_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 51 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   Age                                1470 non-null   int64
 1   Attrition                          1470 non-null   int64
 2   DailyRate                          1470 non-null   int64
 3   DistanceFromHome                   1470 non-null   int64
 4   Education                          1470 non-null   int64
 5   EnvironmentSatisfaction            1470 non-null   int64
 6   HourlyRate                         1470 non-null   int64
 7   JobInvolvement                     1470 non-null   int64
 8   JobLevel                           1470 non-null   int64
 9   JobSatisfaction                    1470 non-null   int64
 10  MonthlyIncome                      1470 non-null   int64
 11  MonthlyRate                        1470 non-null   int64
 12  NumCompaniesWorked                 1470 non-null   int64
 13  OverTime                           1470 non-null   int64
 14  PercentSalaryHike                  1470 non-null   int64
 15  PerformanceRating                  1470 non-null   int64
 16  RelationshipSatisfaction           1470 non-null   int64
 17  StockOptionLevel                   1470 non-null   int64
 18  TotalWorkingYears                  1470 non-null   int64
 19  TrainingTimesLastYear              1470 non-null   int64
 20  WorkLifeBalance                    1470 non-null   int64
 21  YearsAtCompany                     1470 non-null   int64
 22  YearsInCurrentRole                 1470 non-null   int64
 23  YearsSinceLastPromotion            1470 non-null   int64
 24  YearsWithCurrManager               1470 non-null   int64
 25  Department_Human Resources         1470 non-null   uint8
 26  Department_Research & Development  1470 non-null   uint8
 27  Department_Sales                   1470 non-null   uint8
 28  BusinessTravel_Non-Travel          1470 non-null   uint8
 29  BusinessTravel_Travel_Frequently   1470 non-null   uint8
 30  BusinessTravel_Travel_Rarely       1470 non-null   uint8
 31  EducationField_Human Resources     1470 non-null   uint8
 32  EducationField_Life Sciences       1470 non-null   uint8
 33  EducationField_Marketing           1470 non-null   uint8
 34  EducationField_Medical             1470 non-null   uint8
 35  EducationField_Other               1470 non-null   uint8
 36  EducationField_Technical Degree    1470 non-null   uint8
 37  Gender_Female                      1470 non-null   uint8
 38  Gender_Male                        1470 non-null   uint8
 39  JobRole_Healthcare Representative  1470 non-null   uint8
 40  JobRole_Human Resources            1470 non-null   uint8
 41  JobRole_Laboratory Technician      1470 non-null   uint8
 42  JobRole_Manager                    1470 non-null   uint8
 43  JobRole_Manufacturing Director     1470 non-null   uint8
 44  JobRole_Research Director          1470 non-null   uint8
 45  JobRole_Research Scientist         1470 non-null   uint8
 46  JobRole_Sales Executive            1470 non-null   uint8
 47  JobRole_Sales Representative       1470 non-null   uint8
 48  MaritalStatus_Divorced             1470 non-null   uint8
 49  MaritalStatus_Married              1470 non-null   uint8
 50  MaritalStatus_Single               1470 non-null   uint8
dtypes: int64(25), uint8(26)
memory usage: 324.6 KB
```

In [13]:

```
# X and y variable
X = employee_df.drop(['Attrition'],axis=1)
y = employee_df['Attrition']
```

In [14]:

```
#scaling
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X = scaler.fit_transform(X)
```

In [15]:

```
#splitting 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)
X_train.shape
```

Out[15]:

```
(1102, 50)
```

In [16]:

```
# LogisticRegression ML training
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score

model = LogisticRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
```

In [17]:

```
# comparing predicted vs actual value
data_pred = pd.DataFrame(y_pred,columns=['Predicted Value'])
data_pred['Actual Value'] = y_test.values
data_pred
```

Out[17]:

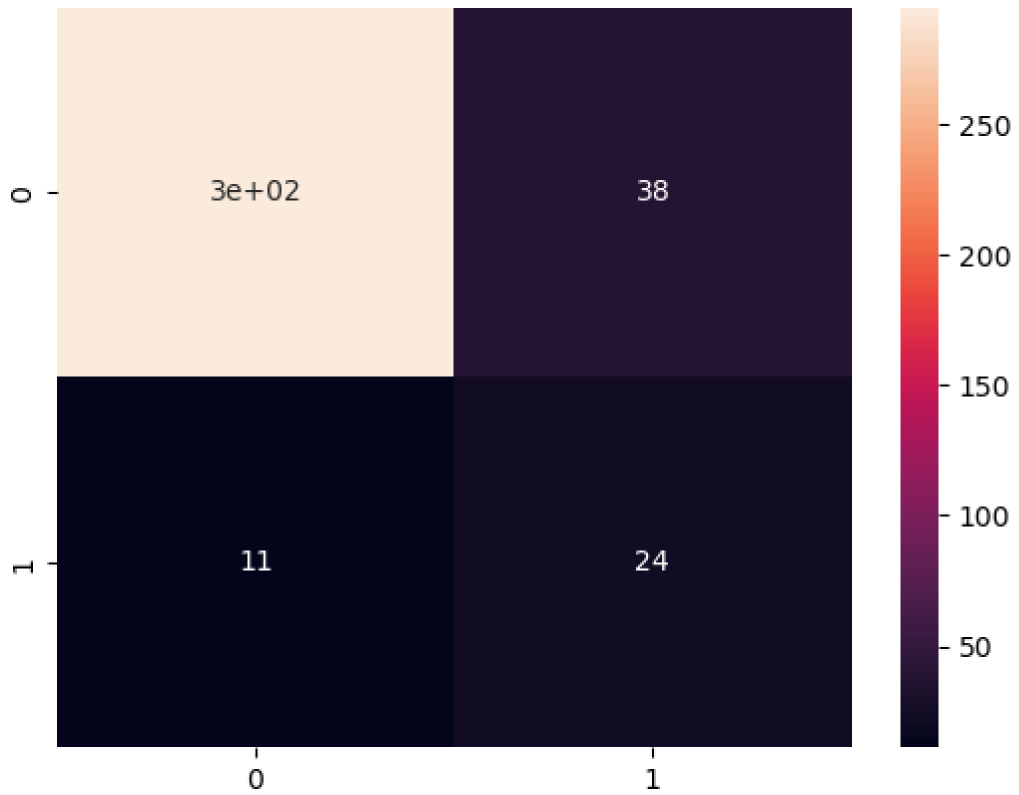
	Predicted Value	Actual Value
0	0	0
1	0	0
2	0	0
3	0	0
4	0	0
...	...	...
363	0	1
364	0	0
365	0	0
366	0	1
367	0	0

368 rows × 2 columns

In [18]:

```
#Model Evaluation
from sklearn.metrics import confusion_matrix, classification_report

print("Accuracy {:.2f} %".format(accuracy_score(y_pred, y_test)*100))
# Testing Set Performance
cm = confusion_matrix(y_pred, y_test)
sns.heatmap(cm, annot=True)

print('\n',classification_report(y_test, y_pred))

```

```
Accuracy 86.68 %

              precision    recall  f1-score   support

           0       0.89      0.96      0.92       306
           1       0.69      0.39      0.49        62

    accuracy                           0.87       368
   macro avg       0.79      0.68      0.71       368
weighted avg       0.85      0.87      0.85       368
```



In [19]:

```
#deep learnig tensorflow
import tensorflow as tf
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(units=500, activation='relu', input_shape=(50,)))
model.add(tf.keras.layers.Dense(units=500, activation='relu'))
model.add(tf.keras.layers.Dense(units=500, activation='relu'))
model.add(tf.keras.layers.Dense(units=1, activation='sigmoid'))
model.summary()
model.compile(optimizer='Adam', loss='binary_crossentropy', metrics = ['accuracy'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 dense (Dense)               (None, 500)               25500     
                                                                 
 dense_1 (Dense)             (None, 500)               250500    
                                                                 
 dense_2 (Dense)             (None, 500)               250500    
                                                                 
 dense_3 (Dense)             (None, 1)                 501       
                                                                 
=================================================================
Total params: 527,001
Trainable params: 527,001
Non-trainable params: 0
_________________________________________________________________
```

In [20]:

```
#training model
epochs_hist = model.fit(X_train, y_train, epochs = 100, batch_size = 50)
```

```
Epoch 1/100
23/23 [==============================] - 1s 7ms/step - loss: 0.4387 - a
ccuracy: 0.8140
Epoch 2/100
23/23 [==============================] - 0s 7ms/step - loss: 0.3497 - a
ccuracy: 0.8548
Epoch 3/100
23/23 [==============================] - 0s 6ms/step - loss: 0.3032 - a
ccuracy: 0.8902
Epoch 4/100
23/23 [==============================] - 0s 7ms/step - loss: 0.2901 - a
ccuracy: 0.8848
Epoch 5/100
23/23 [==============================] - 0s 6ms/step - loss: 0.2556 - a
ccuracy: 0.8984
Epoch 6/100
23/23 [==============================] - 0s 6ms/step - loss: 0.2123 - a
ccuracy: 0.9211
Epoch 7/100
```

In [21]:

```
y_pred = model.predict(X_test)
y_pred = y_pred>0.5
y_pred

```

```
12/12 [==============================] - 0s 4ms/step
```

Out[21]:

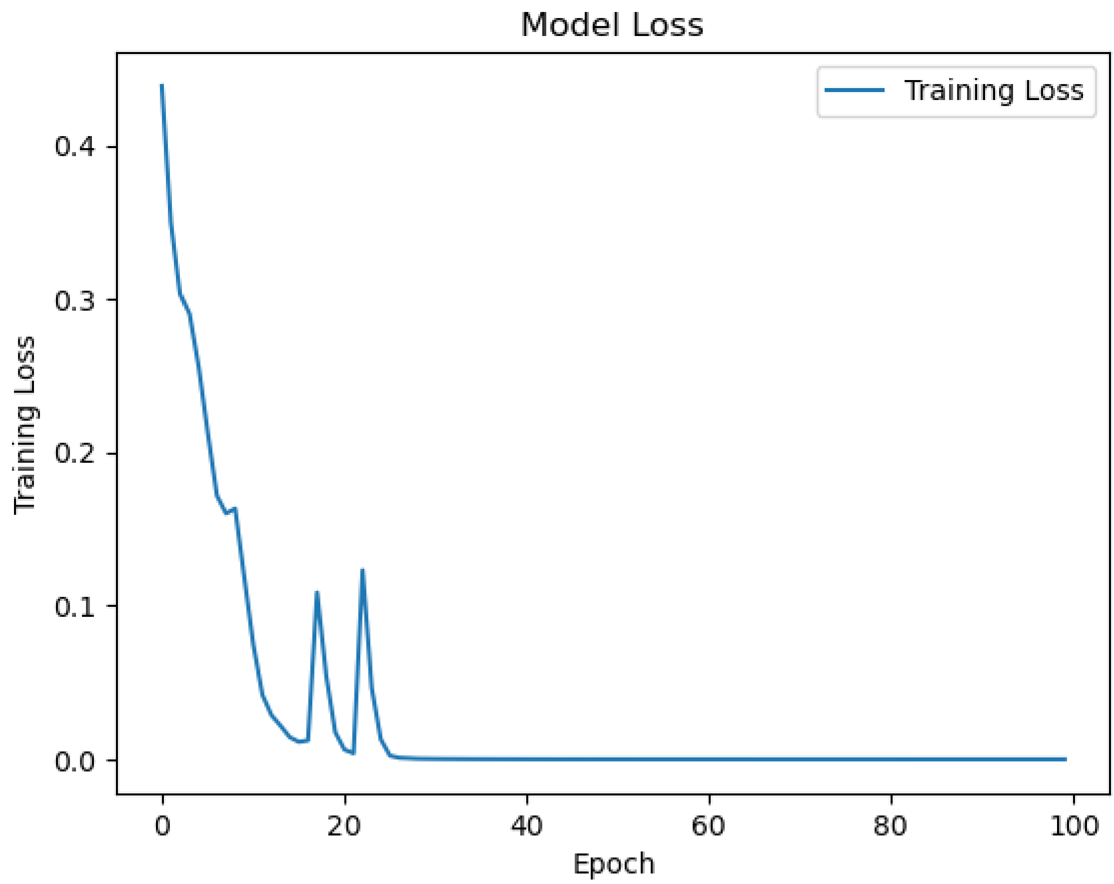
```
array([[False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
       [False],
```

In [22]:

```
plt.plot(epochs_hist.history['loss'])
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Training Loss')
plt.legend(['Training Loss'])
```

Out[22]:

```
<matplotlib.legend.Legend at 0x1c9e6267310>
```

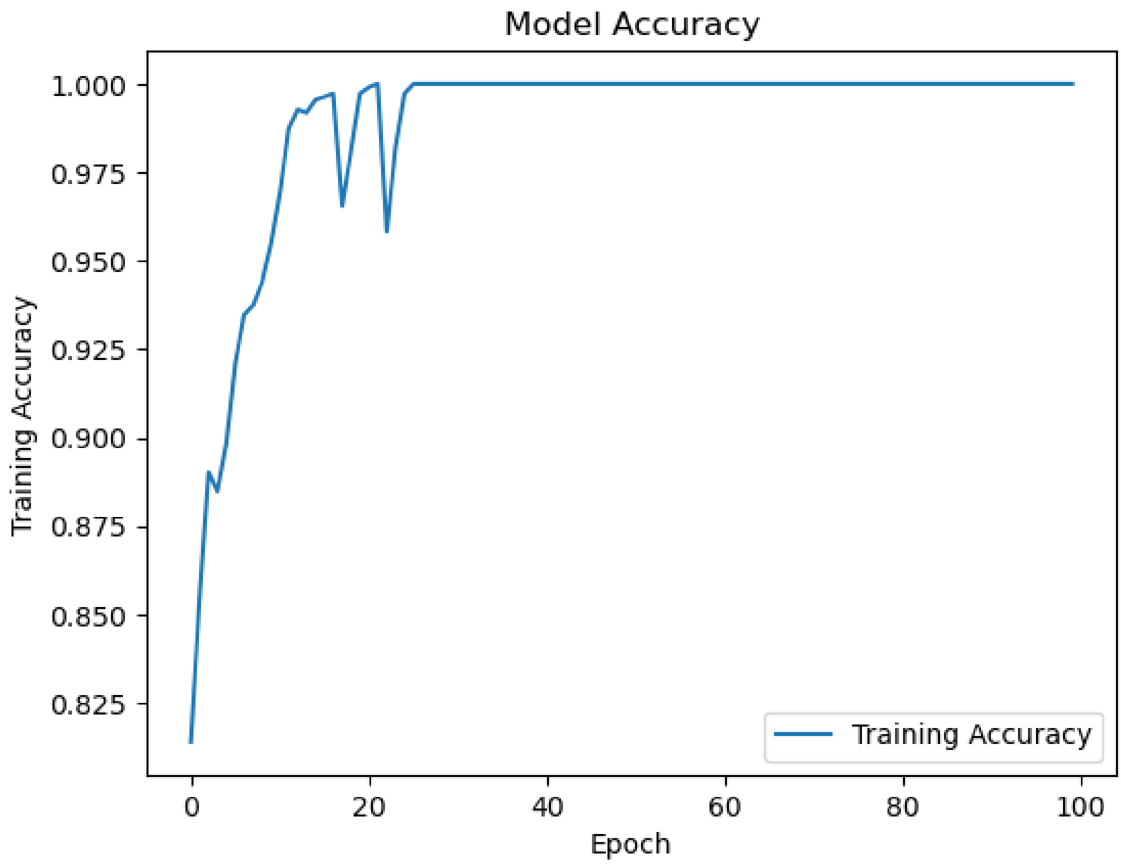


In [23]:

```
plt.plot(epochs_hist.history['accuracy'])
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Training Accuracy')
plt.legend(['Training Accuracy'])
```

Out[23]:

```
<matplotlib.legend.Legend at 0x1c9e6468d90>
```

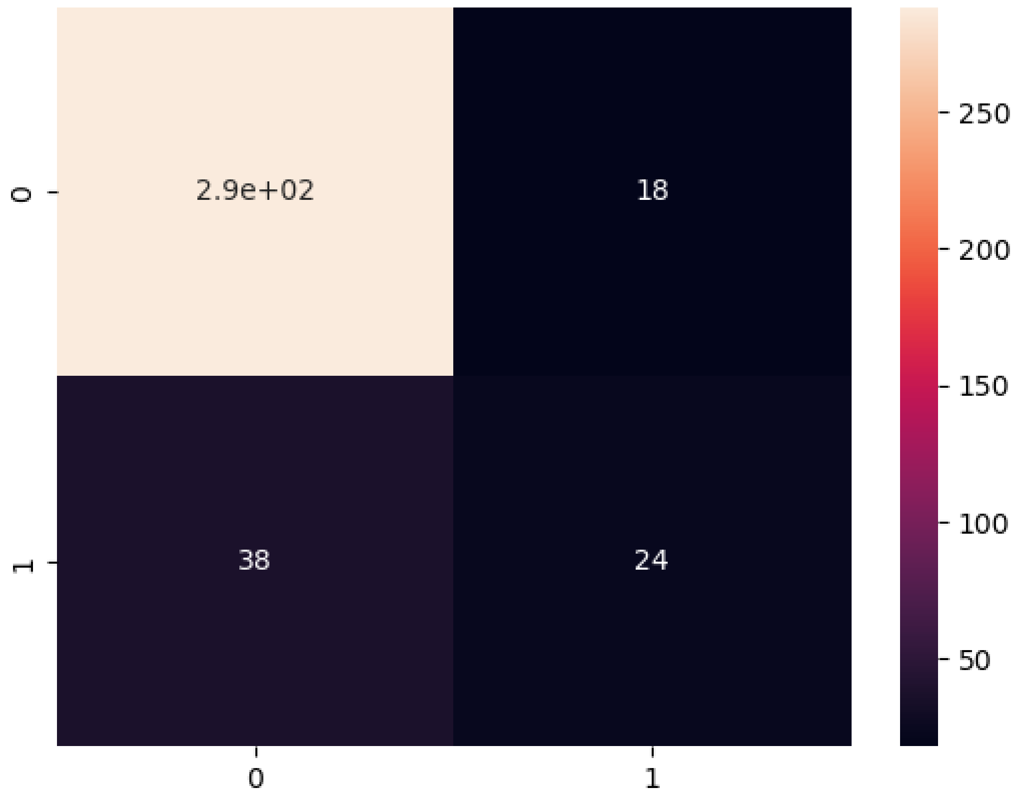


In [24]:

```
#confusion matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True)

```

Out[24]:

<AxesSubplot: >



In [25]:

```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.88      0.94      0.91       306
           1       0.57      0.39      0.46        62

    accuracy                           0.85       368
   macro avg       0.73      0.66      0.69       368
weighted avg       0.83      0.85      0.84       368
```

In [ ]:

```

```