In [1]:

```
# Problem
'''
> theres HR company going to hire new candidate
> candidate has told them his previous salary as reg. manager is 160k/yr
> to check wether he's telling the truth by checking dataset available
> based on dataset only top 10 pos with their salaries been mention
> we have found theres non linear relationship between Pos lvls and Salaries
> goal is to build a Bluffing detector regression model
> we will predict the output for lvl 6.5 bcs the candidate has 4++ yrs exp
> + exp as regional manager, must be somewhere lvls 7 & 6
'''
import numpy as nm
import matplotlib.pyplot as mtp
import pandas as pd
```

In [2]:

```
data_set = pd.read_csv('C:\\Users\\Apeh\\Desktop\\CODE\\DATASET\\Position_Salaries.csv'
data_set.head()
```

Out[2]:

	Position	Level	Salary
0	Business Analyst	1	45000
1	Junior Consultant	2	50000
2	Senior Consultant	3	60000
3	Manager	4	80000
4	Country Manager	5	110000

In [3]:

```
# Dependent & Independent Variables
x = data_set.Level.values[:,nm.newaxis]
x
```

Out[3]:

```
array([[ 1],
       [ 2],
       [ 3],
       [ 4],
       [ 5],
       [ 6],
       [ 7],
       [ 8],
       [ 9],
       [10]], dtype=int64)
```

In [4]:

```
y = data_set.Salary.values
y
```

Out[4]:

```
array([  45000,   50000,   60000,   80000,  110000,  150000,  200000,
        300000,  500000, 1000000], dtype=int64)
```

In [6]:

```
# Apply linear regression model
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(x,y)
```

```
Out[6]: LinearRegression()
```

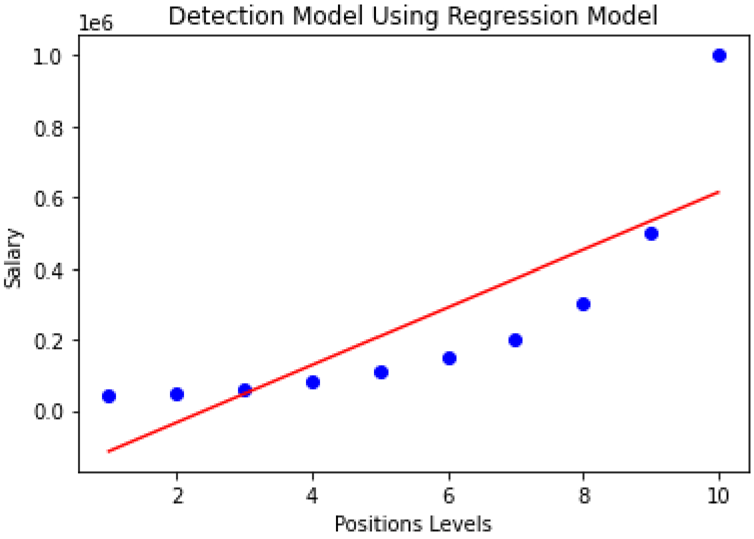
```
In [8]:
# Fitting the polynomial regression dataset
from sklearn.preprocessing import PolynomialFeatures
p_reg = PolynomialFeatures(degree=2)
x_po = p_reg.fit_transform(x)
print(x_po)
```

```
[[  1.   1.   1.]
 [  1.   2.   4.]
 [  1.   3.   9.]
 [  1.   4.  16.]
 [  1.   5.  25.]
 [  1.   6.  36.]
 [  1.   7.  49.]
 [  1.   8.  64.]
 [  1.   9.  81.]
 [  1.  10. 100.]]
```

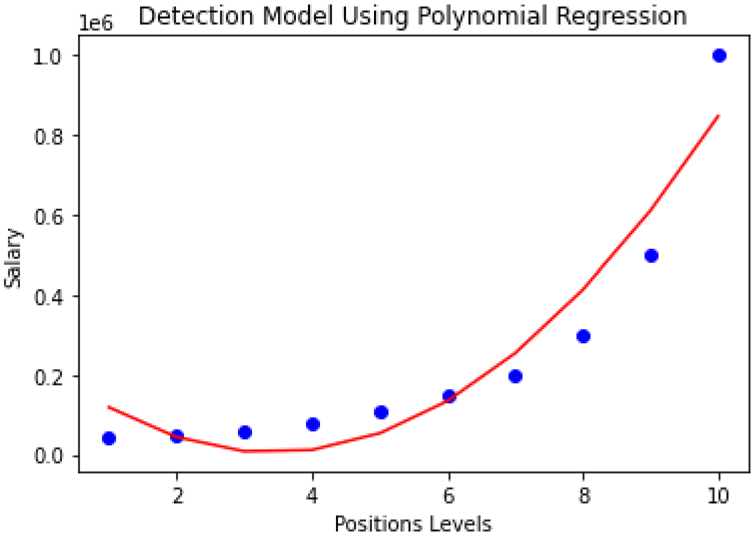
```
In [9]:
lin_reg2 = LinearRegression()
lin_reg2.fit(x_po,y)
```

```
Out[9]: LinearRegression()
```

```
In [10]:
# Visualizing the result for Linear Reg model
mtp.scatter(x,y,color='blue')
mtp.plot(x,lin_reg.predict(x),color='red')
mtp.title('Detection Model Using Regression Model')
mtp.xlabel('Positions Levels')
mtp.ylabel('Salary')
mtp.show()
print('\n')
```



```
In [14]:
# Visualising the result for Polynomial Regression Model
mtp.scatter(x,y,color='blue')
mtp.plot(x,lin_reg2.predict(p_reg.fit_transform(x)),color='red')
mtp.title('Detection Model Using Polynomial Regression')
mtp.xlabel('Positions Levels')
mtp.ylabel('Salary')
mtp.show()
```



In [12]:

```
# Adjust the Degree = 3
p_reg3 = PolynomialFeatures(degree=3)
x_po3 = p_reg3.fit_transform(x)
lin_reg3 = LinearRegression()
lin_reg3.fit(x_po3,y)
```

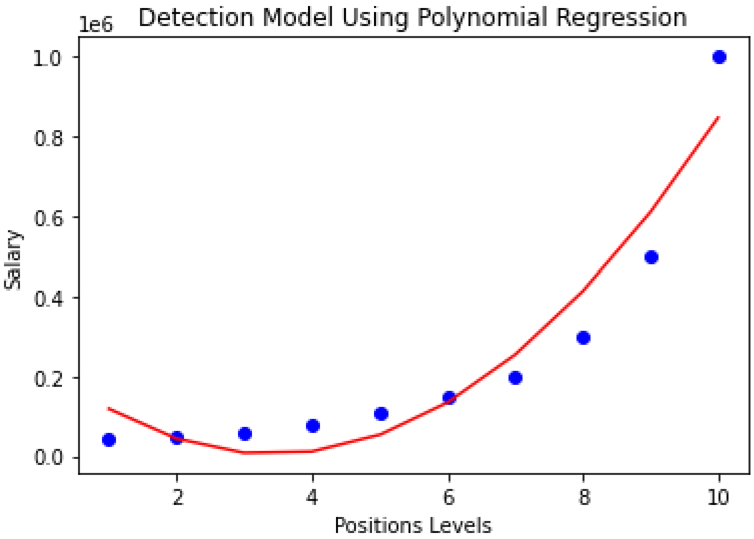
Out[12]:

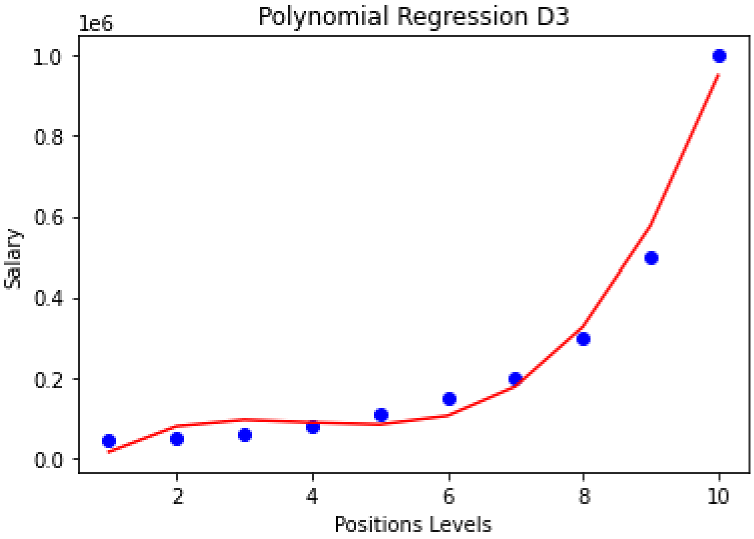
```
LinearRegression()
```

In [13]:

```
# Visualising the result for Polynomial Regression Model
mtp.scatter(x,y,color='blue')
mtp.plot(x,lin_reg2.predict(p_reg.fit_transform(x)),color='red')
mtp.title('Detection Model Using Polynomial Regression')
mtp.xlabel('Positions Levels')
mtp.ylabel('Salary')
mtp.show()
print('\n')

# Degree = 3
mtp.scatter(x,y,color='blue')
mtp.plot(x,lin_reg3.predict(p_reg3.fit_transform(x)),color='red')
mtp.title('Polynomial Regression D3')
mtp.xlabel('Positions Levels')
mtp.ylabel('Salary')
mtp.show()
print('\n')
```



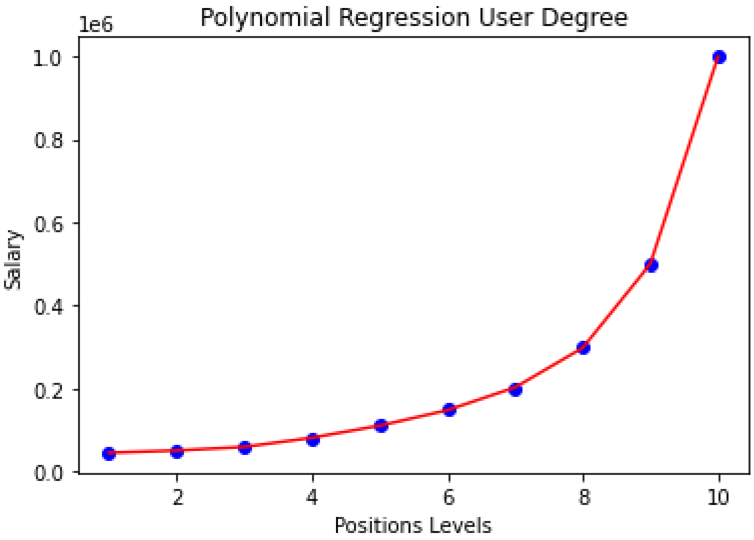


In [15]:

```
# Adjust more Degree = ?
U_deg = int(input('Enter Degree : ', ))
p_regU = PolynomialFeatures(degree=U_deg)
x_poU = p_regU.fit_transform(x)
lin_regU = LinearRegression()
lin_regU.fit(x_poU,y)

mtp.scatter(x,y,color='blue')
mtp.plot(x,lin_regU.predict(p_regU.fit_transform(x)),color='red')
mtp.title('Polynomial Regression User Degree')
mtp.xlabel('Positions Levels')
mtp.ylabel('Salary')
mtp.show()
```

```
Enter Degree : 7
```



In [18]:

```
# Making the prediction with Linear Regression Model
l_pred = lin_reg.predict([[6.5]])
print(l_pred)   # not accurate, not tally with candidate statement
```

```
[330378.78787879]
```

In [19]:

```
# Making the prediction with Polynomial Regression Model
p_pred = lin_regU.predict(p_regU.fit_transform([[6.5]]))
print(p_pred)   # 170k closer to the 160k value given
```

```
[172108.37620211]
```