IMPORTING LIBRARIES

In [26]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [27]:
```
data = pd.read_csv(r"D:\DataScienceGenai\Polynomial_regression\Polynomial regressio
data
```

Out[27]:

	Position	Level	Salary
0	Jr Software Engineer	1	45000
1	Sr Software Engineer	2	50000
2	Team Lead	3	60000
3	Manager	4	80000
4	Sr manager	5	110000
5	Region Manager	6	150000
6	AVP	7	200000
7	VP	8	300000
8	CTO	9	500000
9	CEO	10	1000000

In [28]:
```
X = data.iloc[:,1:2].values
y = data.iloc[:,2].values
```

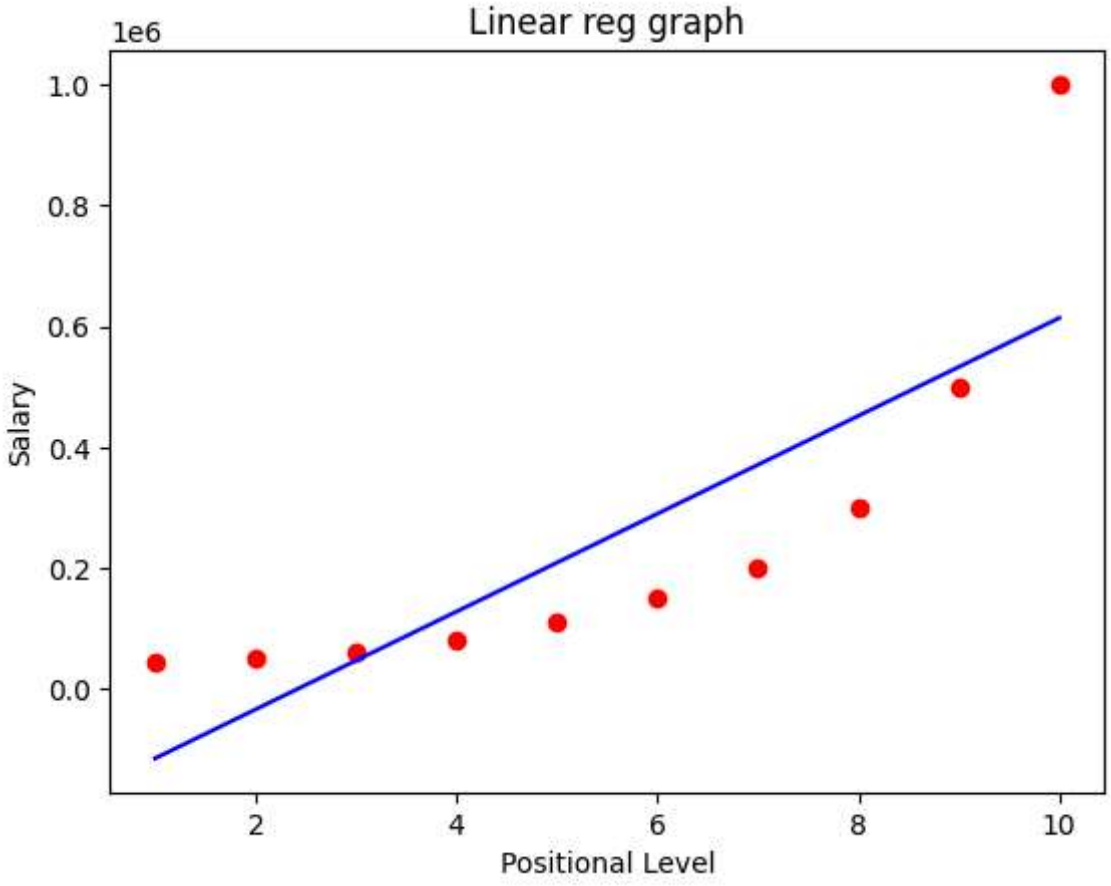
In [29]:
```
from sklearn.linear_model import LinearRegression
lin_reg = LinearRegression()
lin_reg.fit(X,y)
```

Out[29]:

LinearRegression

```
LinearRegression()
```

In [30]:
```
plt.scatter(X,y, color = 'red')
plt.plot(X,lin_reg.predict(X),color = 'blue')
plt.title("Linear reg graph")
plt.xlabel("Positional Level")
plt.ylabel("Salary")
plt.show()
```



In [31]:
```
lin_model_pred = lin_reg.predict(([[6.5]]))
print(f"Linear Regression Prediction for 6.5: {lin_model_pred}")
```

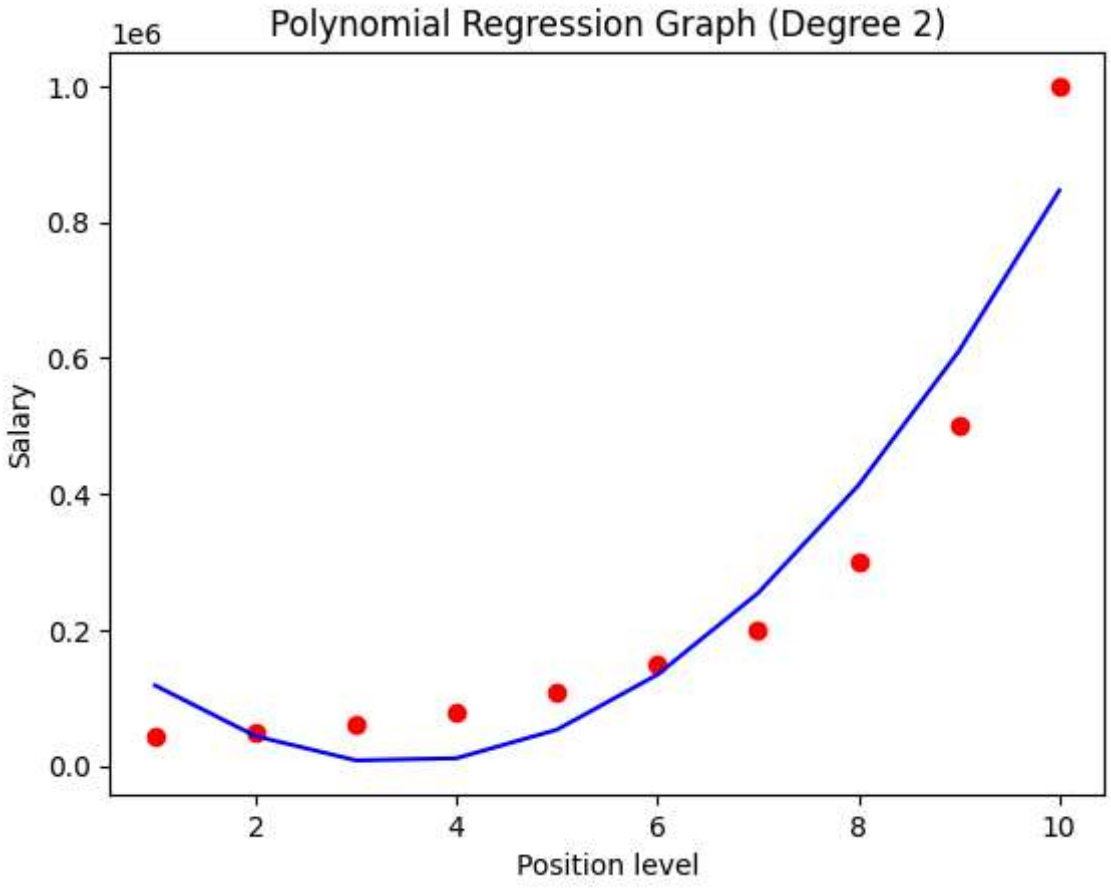
```
Linear Regression Prediction for 6.5: [330378.78787879]
```

In [32]:
```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=2)
X_poly = poly_reg.fit_transform(X)
```

In [33]:
```
# Create and train the Polynomial Regression model
poly_regressor = LinearRegression()
poly_regressor.fit(X_poly, y)
```

Out[33]:
```
LinearRegression
LinearRegression()
```

In [34]:
```
plt.scatter(X,y, color = 'red')
plt.plot(X,poly_regressor.predict(X_poly), color = 'blue') # Use X_poly for predict
plt.title("Polynomial Regression Graph (Degree 2)")
plt.xlabel("Position level")
plt.ylabel("Salary")
plt.show()
```



In [35]:
```
poly_model_pred = poly_regressor.predict(poly_reg.transform([[6.5]])) # Use poly_re
print(f"Polynomial Regression Prediction for 6.5: {poly_model_pred}")
```

```
Polynomial Regression Prediction for 6.5: [189498.10606061]
```

In [36]:
```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=3)
X_poly = poly_reg.fit_transform(X)
```

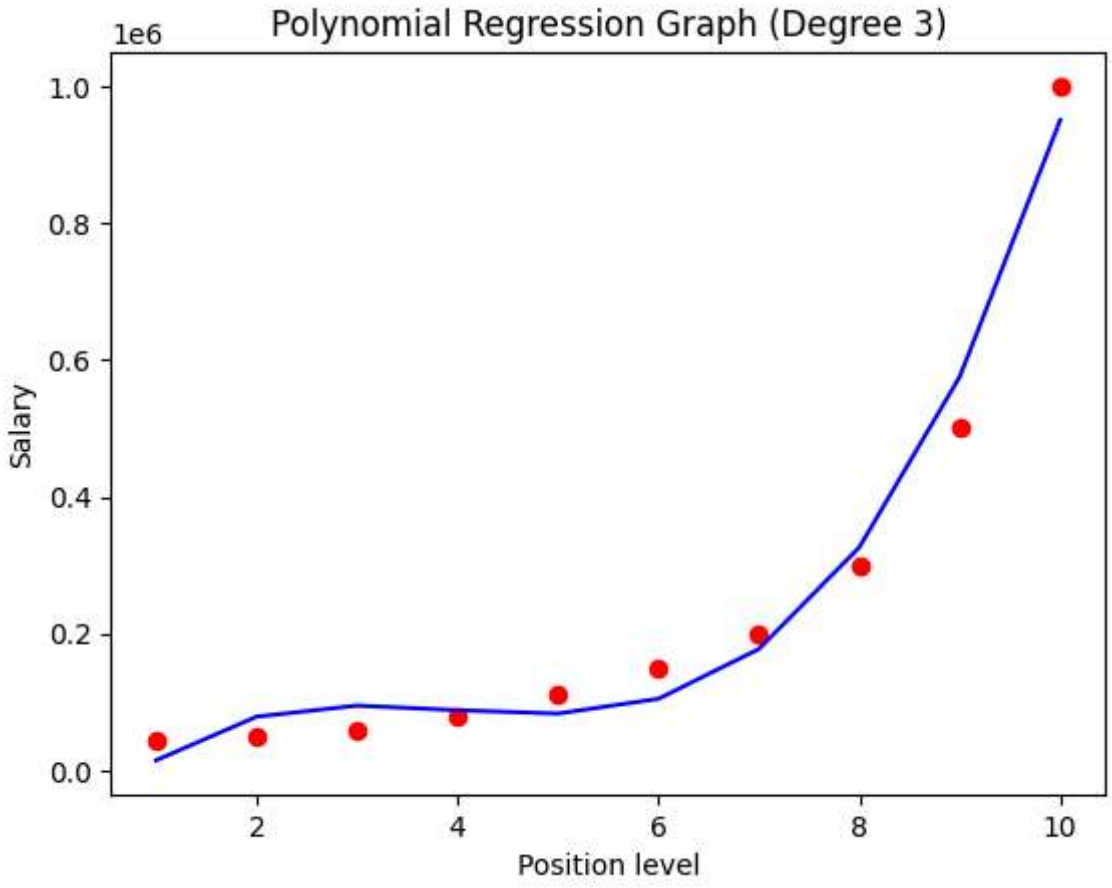
In [37]:
```
# Create and train the Polynomial Regression model
poly_regressor = LinearRegression()
poly_regressor.fit(X_poly, y)
```

Out[37]:

▾ LinearRegression

```
LinearRegression()
```

In [38]:
```
plt.scatter(X,y, color = 'red')
plt.plot(X,poly_regressor.predict(X_poly), color = 'blue') # Use X_poly for predict
plt.title("Polynomial Regression Graph (Degree 3)")
plt.xlabel("Position level")
plt.ylabel("Salary")
plt.show()
```



In [39]:
```
poly_model_pred = poly_regressor.predict(poly_reg.transform([[6.5]])) # Use poly_re
print(f"Polynomial Regression Prediction for 6.5: {poly_model_pred}")
```

```
Polynomial Regression Prediction for 6.5: [133259.46969697]
```

In [40]:
```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=4)
X_poly = poly_reg.fit_transform(X)
```

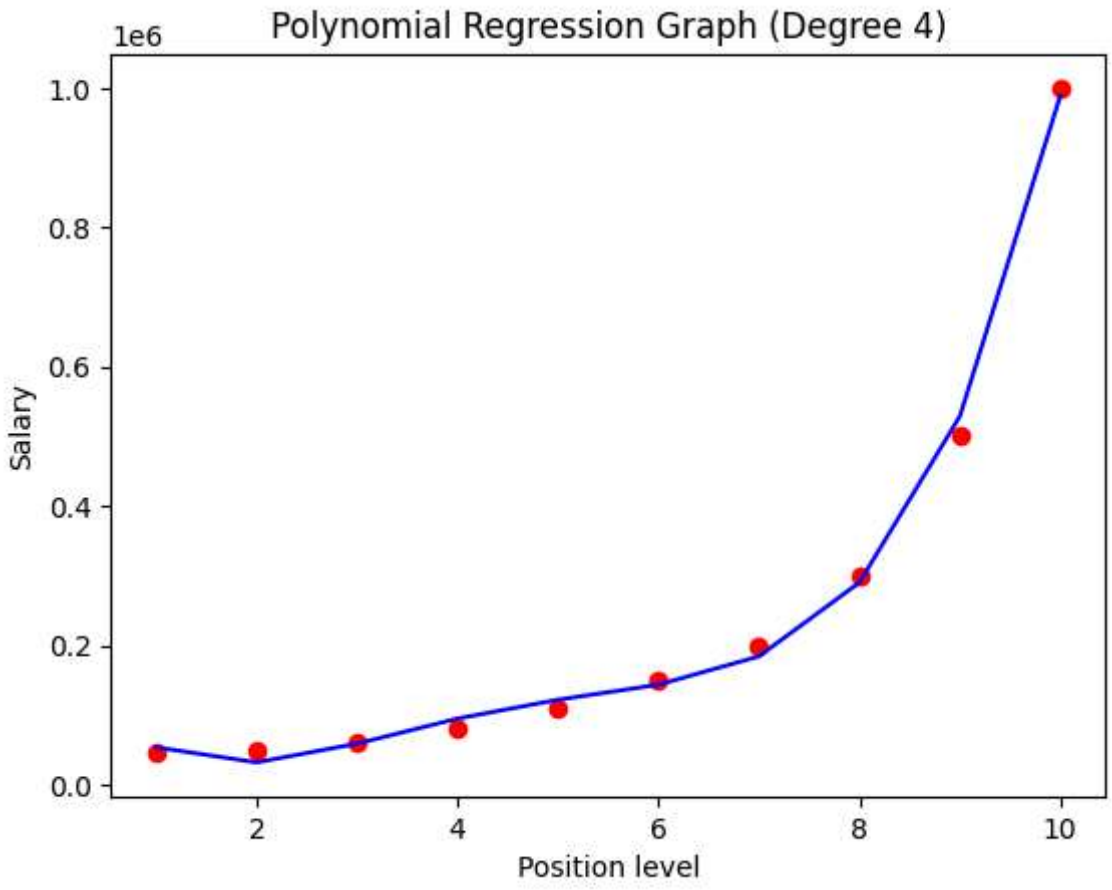
In [41]:
```
# Create and train the Polynomial Regression model
poly_regressor = LinearRegression()
poly_regressor.fit(X_poly, y)
```

Out[41]:

▾ LinearRegression

```
LinearRegression()
```

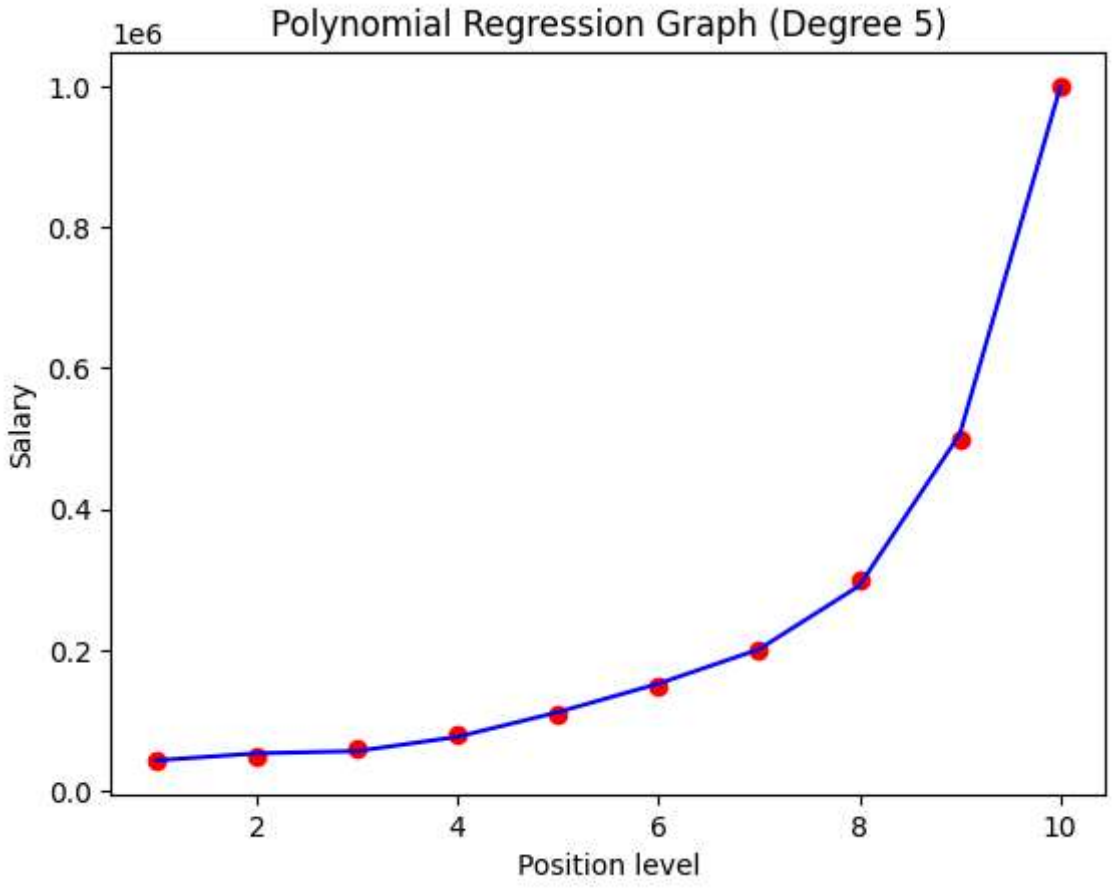
In [42]:
```
plt.scatter(X,y, color = 'red')
plt.plot(X,poly_regressor.predict(X_poly), color = 'blue') # Use X_poly for predict
plt.title("Polynomial Regression Graph (Degree 4)")
plt.xlabel("Position level")
plt.ylabel("Salary")
plt.show()

poly_model_pred = poly_regressor.predict(poly_reg.transform([[6.5]])) # Use poly_re
print(f"Polynomial Regression Prediction for 6.5: {poly_model_pred}")
```




```
Polynomial Regression Prediction for 6.5: [158862.45265155]
```

In [43]:

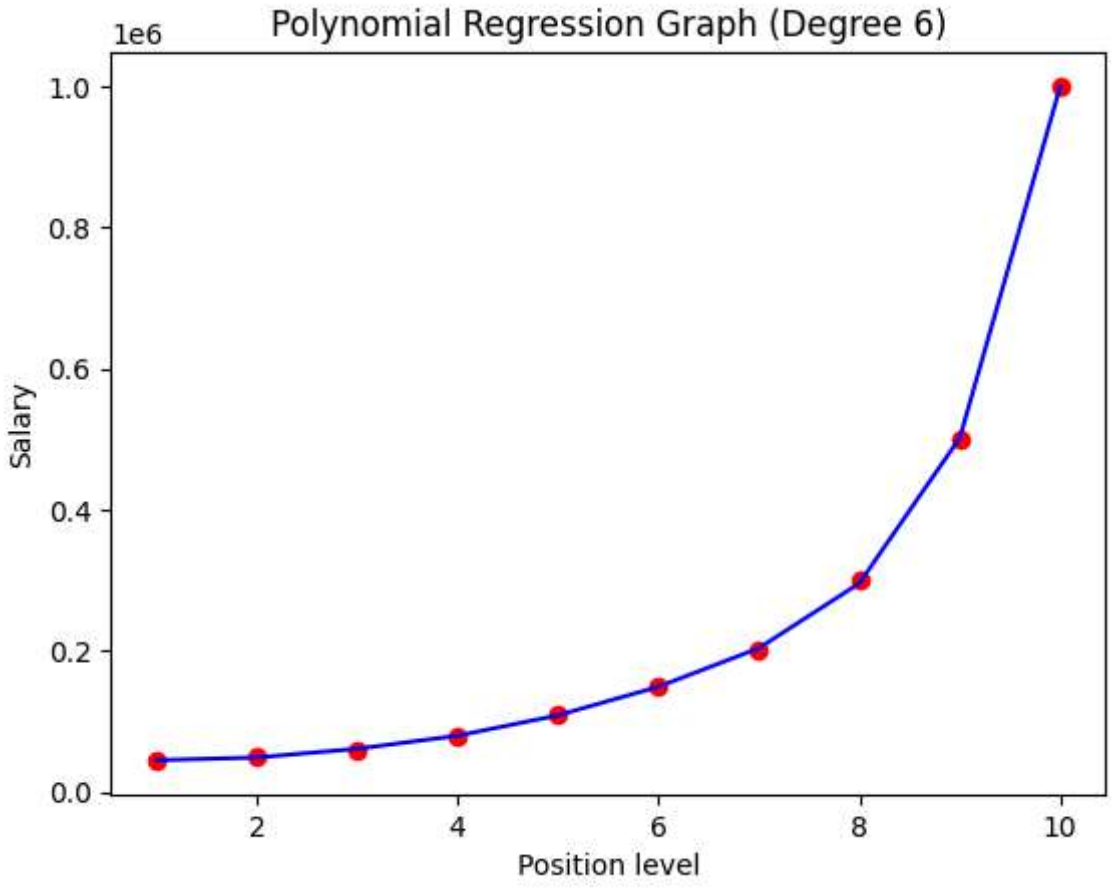
```python
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=5)
X_poly = poly_reg.fit_transform(X)

# Create and train the Polynomial Regression model
poly_regressor = LinearRegression()
poly_regressor.fit(X_poly, y)

plt.scatter(X,y, color = 'red')
plt.plot(X,poly_regressor.predict(X_poly), color = 'blue') # Use X_poly for predict
plt.title("Polynomial Regression Graph (Degree 5)")
plt.xlabel("Position level")
plt.ylabel("Salary")
plt.show()
```



In [44]:
```
poly_model_pred = poly_regressor.predict(poly_reg.transform([[6.5]])) # Use poly_re
print(f"Polynomial Regression Prediction for 6.5: {poly_model_pred}")
```

```
Polynomial Regression Prediction for 6.5: [174878.07765173]
```

In [45]:
```
from sklearn.preprocessing import PolynomialFeatures
poly_reg = PolynomialFeatures(degree=6)
X_poly = poly_reg.fit_transform(X)

# Create and train the Polynomial Regression model
poly_regressor = LinearRegression()
poly_regressor.fit(X_poly, y)

plt.scatter(X,y, color = 'red')
plt.plot(X,poly_regressor.predict(X_poly), color = 'blue') # Use X_poly for predict
plt.title("Polynomial Regression Graph (Degree 6)")
plt.xlabel("Position level")
plt.ylabel("Salary")
plt.show()

poly_model_pred = poly_regressor.predict(poly_reg.transform([[6.5]])) # Use poly_re
print(f"Polynomial Regression Prediction for 6.5: {poly_model_pred}")
```



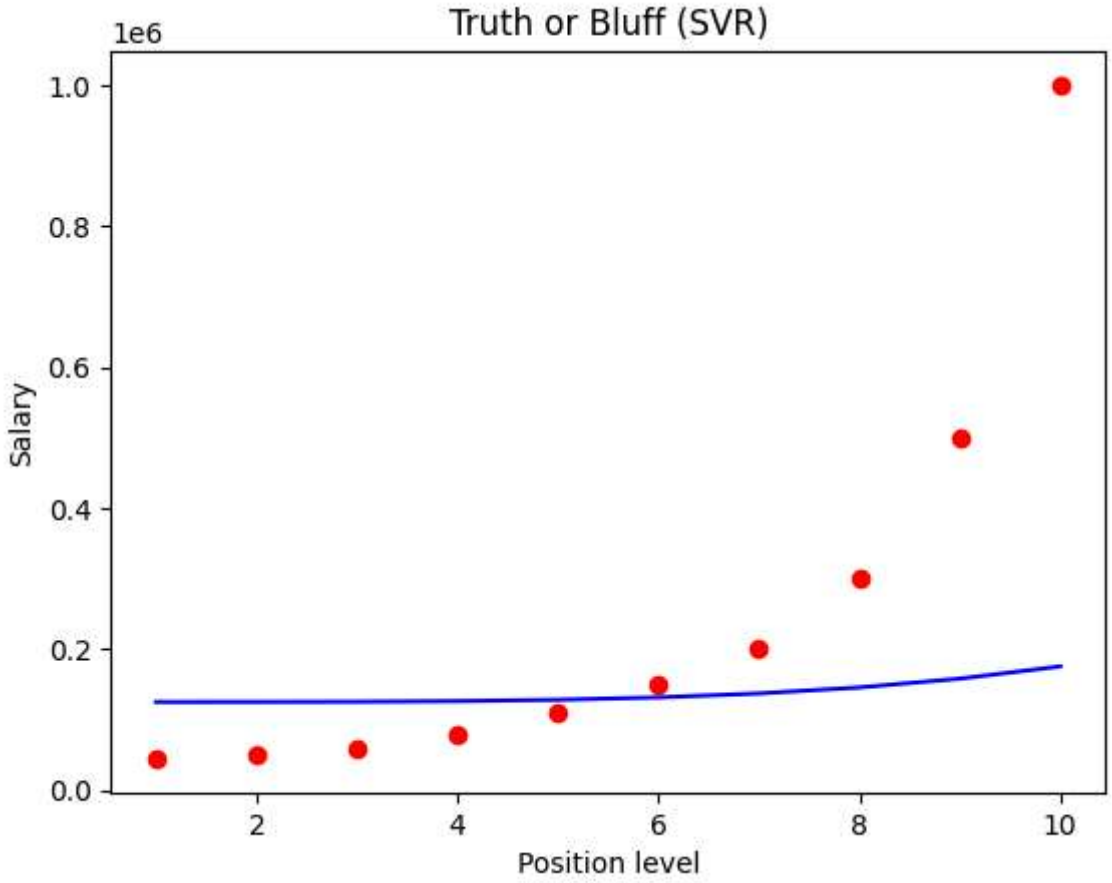

```
Polynomial Regression Prediction for 6.5: [174192.81930603]
```

In [46]:

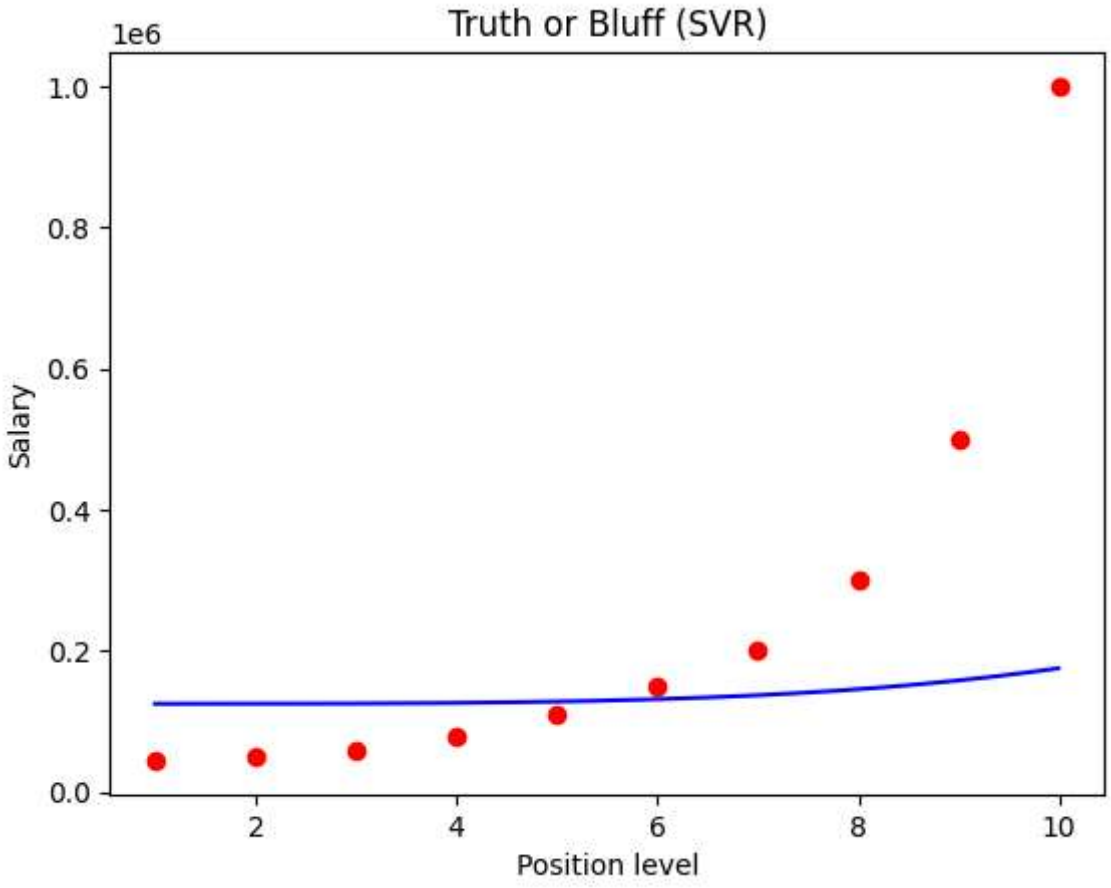
```python
# svm model
from sklearn.svm import SVR
svr_regressor = SVR(kernel='poly',degree = 5,gamma = 'scale' )
svr_regressor.fit(X,y)

svr_model_pred = svr_regressor.predict([[6.5]])
print(svr_model_pred)


# Fitting SVR to the dataset
from sklearn.svm import SVR
regressor = SVR(kernel = 'poly',degree = 4)
regressor.fit(X, y)

y_pred_svr = regressor.predict([[6.5]])


# Visualising the SVR results
plt.scatter(X, y, color = 'red')
plt.plot(X, regressor.predict(X), color = 'blue')
plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()


# Visualising the SVR results (for higher resolution and smoother curve)
```

```
X_grid = np.arange(min(X), max(X), 0.01) # choice of 0.01 instead of 0.1 step becau
X_grid = X_grid.reshape((len(X_grid), 1))
plt.scatter(X, y, color = 'red')
plt.plot(X_grid, regressor.predict(X_grid), color = 'blue')
plt.title('Truth or Bluff (SVR)')
plt.xlabel('Position level')
plt.ylabel('Salary')
plt.show()
```

```
[164079.01344549]
```




```
C:\Users\mohap\AppData\Local\Temp\ipykernel_17244\3788301405.py:28: DeprecationWarni
ng: Conversion of an array with ndim > 0 to a scalar is deprecated, and will error i
n future. Ensure you extract a single element from your array before performing this
operation. (Deprecated NumPy 1.25.)
  X_grid = np.arange(min(X), max(X), 0.01) # choice of 0.01 instead of 0.1 step beca
use the data is feature scaled
```



In [47]:
```
# knn model
from sklearn.neighbors import KNeighborsRegressor
knn_reg_model = KNeighborsRegressor(n_neighbors=4, weights='distance', p=1, metric=
knn_reg_model.fit(X,y)

knn_reg_pred = knn_reg_model.predict([[6.5]])
print(knn_reg_pred)
1
```

```
[182500.]
```

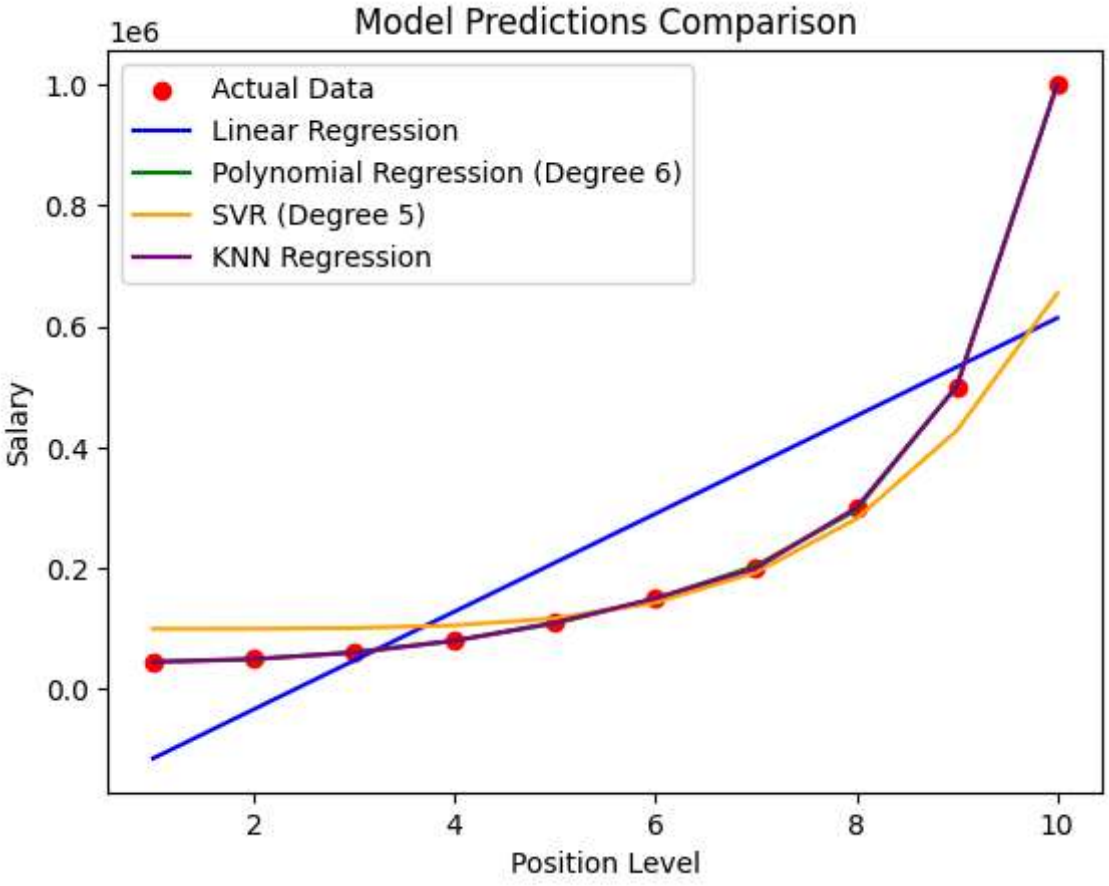
Out[47]: 1

In [48]:
```
# knn model
from sklearn.neighbors import KNeighborsRegressor
knn_reg_model = KNeighborsRegressor(n_neighbors=5, weights='distance', p=2,gamma= '
knn_reg_model.fit(X,y)

knn_reg_pred = knn_reg_model.predict([[6.5]])
print(knn_reg_pred)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[48], line 3
      1 # knn model
      2 from sklearn.neighbors import KNeighborsRegressor
----> 3 knn_reg_model = KNeighborsRegressor(n_neighbors=5, weights='distance', p=2,g
amma= 'auto')
      4 knn_reg_model.fit(X,y)
      6 knn_reg_pred = knn_reg_model.predict([[6.5]])

TypeError: KNeighborsRegressor.__init__() got an unexpected keyword argument 'gamma'
```

In [ ]:

```python
# Plotting the actual data points
plt.scatter(X, y, color='red', label='Actual Data')

# Plotting Linear Regression predictions
plt.plot(X, lin_reg.predict(X), color='blue', label='Linear Regression')

# Plotting Polynomial Regression predictions
plt.plot(X, poly_regressor.predict(X_poly), color='green', label='Polynomial Regres

# Plotting SVR predictions
plt.plot(X, svr_regressor.predict(X), color='orange', label='SVR (Degree 5)')

# Plotting KNN predictions
plt.plot(X, knn_reg_model.predict(X), color='purple', label='KNN Regression')

# Adding labels and title
plt.title('Model Predictions Comparison')
plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.legend()
plt.show()
```



In [ ]:
```
from sklearn.tree import DecisionTreeRegressor
regrssor_dtr = DecisionTreeRegressor(criterion='absolute_error',splitter='random',r
regrssor_dtr.fit(X, y)
# Create and train the Decision Tree Regressor

# decision tree
```

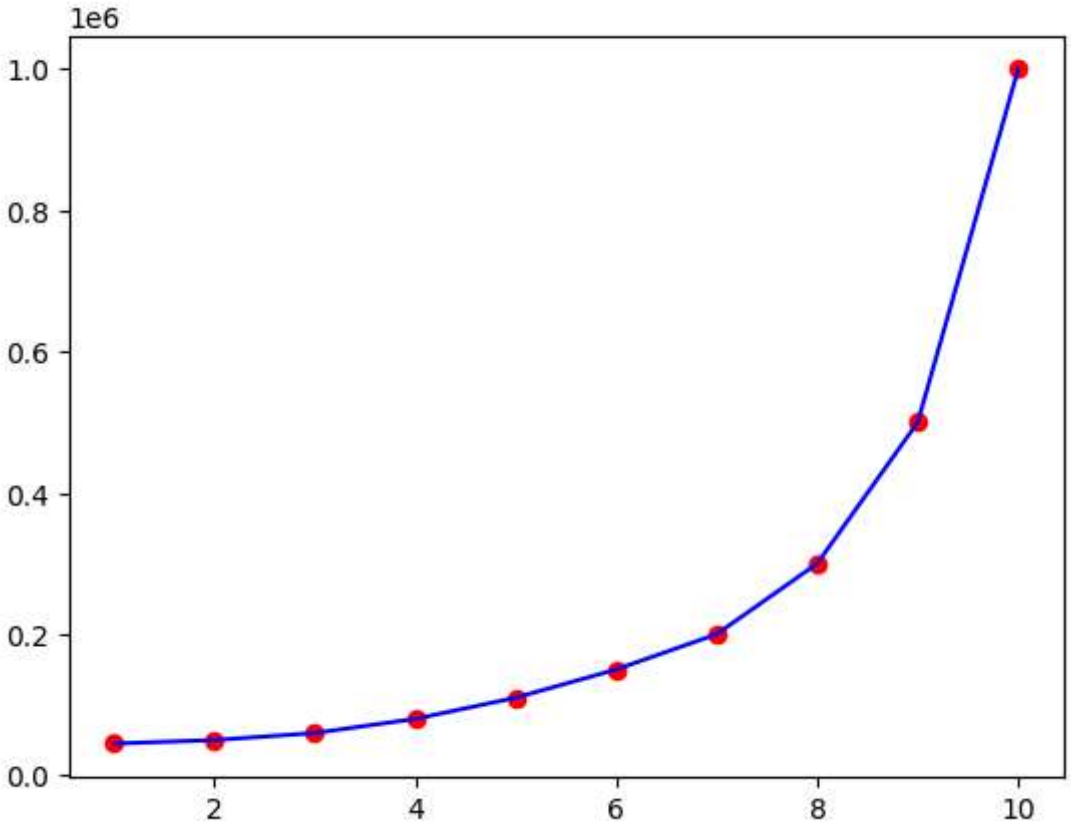
Out[ ]:
```
DecisionTreeRegressor(criterion='absolute_error', random_state=0,
                      splitter='random')
```

In [ ]:
```
y_pred_dtr = regrssor_dtr.predict([[6.5]])
print(y_pred_dtr)
```

```
[200000.]
```

In [ ]:
```
y_pred_svr = regrssor_dtr.predict([[6.5]])
# Visualising the Decision Tree results
plt.scatter(X, y, color = 'red')
plt.plot(X, regrssor_dtr.predict(X), color = 'blue')
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x1b77b2ac080>]
```



In [ ]:
```python
from sklearn.model_selection import GridSearchCV

# Define the parameter grid
param_grid = {
    'max_depth': [None, 5, 10, 15],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

# Perform GridSearchCV
grid_search = GridSearchCV(estimator=regrssor_dtr, param_grid=param_grid, cv=5, sco
grid_search.fit(X, y)

# Best parameters and score
print("Best Parameters:", grid_search.best_params_)
print("Best Score:", -grid_search.best_score_)
```

```
Best Parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2}
Best Score: 56692500000.0
```

In [ ]:
```python
# RANDOM FOREST REGRESSOR
from sklearn.ensemble import RandomForestRegressor
rf_regressor = RandomForestRegressor(n_estimators=100,random_state=0, max_depth=5,
# Create and train the Random Forest Regressor
rf_regressor.fit(X, y)
```

Out[ ]:

```
RandomForestRegressor
RandomForestRegressor(max_depth=5, random_state=0)
```

In [ ]:

```
# Predicting a new result with Random Forest Regression
y_pred_rf = rf_regressor.predict([[6.5]])
print(y_pred_rf)
```

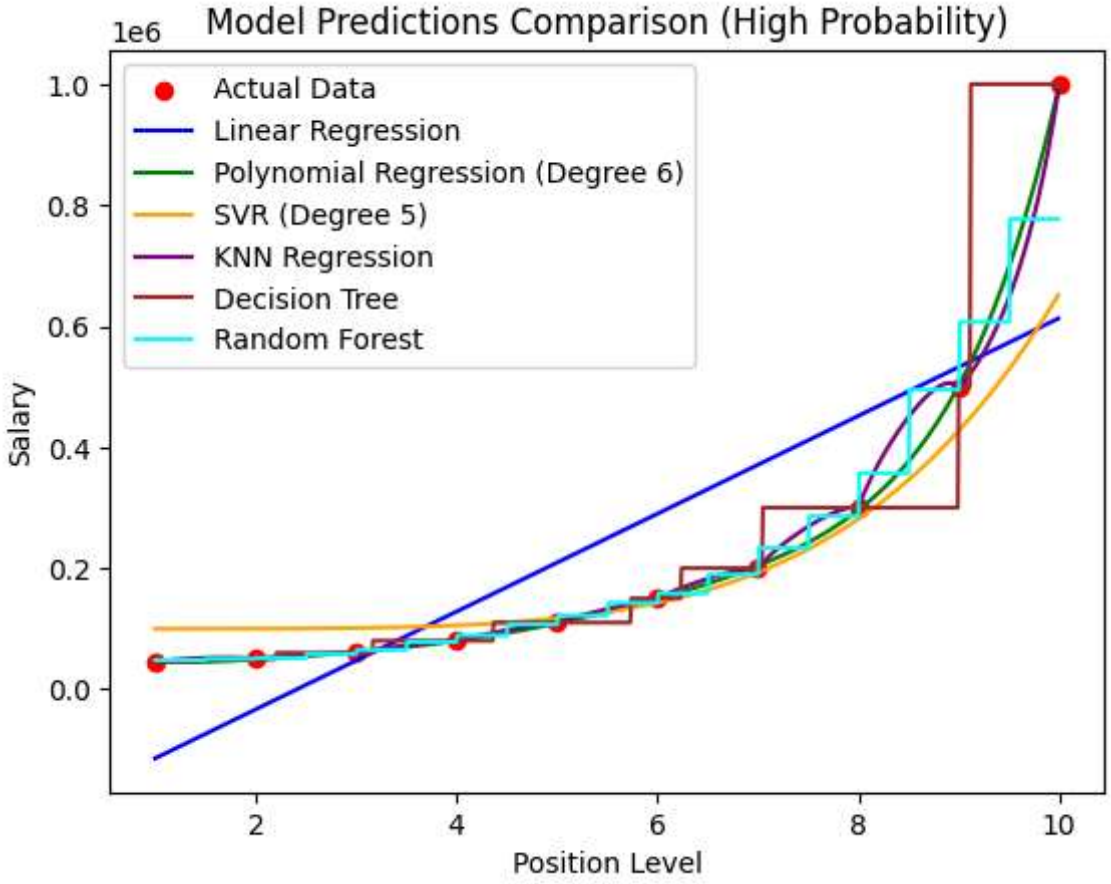
```
[158300.]
```

The time taken by all models mentioned above is as follows:

- **Linear Regression Time**: 0.002021 seconds
- **Polynomial Regression Time**: 0.004050 seconds
- **SVR Time**: 0.002304 seconds
- **KNN Time**: 0.005392 seconds
- **Decision Tree Time**: 0.001779 seconds
- **Random Forest Time**: 0.010905 seconds

In [54]:

```
# Plotting the actual data points
plt.scatter(X, y, color='red', label='Actual Data')

# Plotting Linear Regression predictions
plt.plot(X_grid, lin_reg.predict(X_grid), color='blue', label='Linear Regression')

# Plotting Polynomial Regression predictions
plt.plot(X_grid, poly_regressor.predict(poly_reg.transform(X_grid)), color='green',

# Plotting SVR predictions
plt.plot(X_grid, svr_regressor.predict(X_grid), color='orange', label='SVR (Degree

# Plotting KNN predictions
plt.plot(X_grid, knn_reg_model.predict(X_grid), color='purple', label='KNN Regressi

# Plotting Decision Tree predictions
plt.plot(X_grid, regrssor_dtr.predict(X_grid), color='brown', label='Decision Tree'

# Plotting Random Forest predictions
plt.plot(X_grid, rf_regressor.predict(X_grid), color='cyan', label='Random Forest')

# Adding labels and title
plt.title('Model Predictions Comparison (High Probability)')
plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.legend()
plt.show()
```


Model Predictions Comparison (High Probability)
1e6
1.0
0.8
0.6
0.4
0.2
0.0
Salary
2
4
6
8
10
Position Level
Actual Data
Linear Regression
Polynomial Regression (Degree 6)
SVR (Degree 5)
KNN Regression
Decision Tree
Random Forest