You are provided with historical stock price data, and you suspect that the relationship between time and stock price is non-linear due to market trends. Develop a polynomial regression model to predict future stock prices based on historical time-series data. Ensure that the model effectively captures non-linear trends and provides reliable forecasts

In [60]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [61]:
```
data= pd.read_csv('AAPL.csv')
data
```

Out[61]:

	Date	Open	High	Low	Close	Volume	Dividends	Stock Splits
0	2019-05-31	43.181805	43.613062	42.877969	42.897572	108174400	0.0	0.0
1	2019-06-03	43.027439	43.595909	41.721424	42.463867	161584400	0.0	0.0
2	2019-06-04	42.988224	44.063910	42.762796	44.017353	123872000	0.0	0.0
3	2019-06-05	45.154295	45.328268	44.384898	44.727940	119093600	0.0	0.0
4	2019-06-06	44.860258	45.445882	44.632378	45.384624	90105200	0.0	0.0
...	...	...	...	...	...	...	...	...
521	2021-06-24	134.248856	134.438574	132.731126	133.210419	68711000	0.0	0.0
522	2021-06-25	133.260356	133.689705	132.611319	132.910873	70783700	0.0	0.0
523	2021-06-28	133.210424	135.047667	133.150516	134.578369	62111300	0.0	0.0
524	2021-06-29	134.598343	136.285817	134.149019	136.126053	64556100	0.0	0.0
525	2021-06-30	135.966285	137.204435	135.666731	136.755112	63261400	0.0	0.0

526 rows × 8 columns

In [62]:
```
data.shape
```

Out[62]: (526, 8)

In [63]:
```
# ignore last two columns it is because there is no need
data=data.iloc[:,[0,1,2,3,4,5]]
data
```

Out[63]:

	Date	Open	High	Low	Close	Volume
0	2019-05-31	43.181805	43.613062	42.877969	42.897572	108174400
1	2019-06-03	43.027439	43.595909	41.721424	42.463867	161584400
2	2019-06-04	42.988224	44.063910	42.762796	44.017353	123872000
3	2019-06-05	45.154295	45.328268	44.384898	44.727940	119093600
4	2019-06-06	44.860258	45.445882	44.632378	45.384624	90105200
...	...	...	...	...	...	...
521	2021-06-24	134.248856	134.438574	132.731126	133.210419	68711000
522	2021-06-25	133.260356	133.689705	132.611319	132.910873	70783700
523	2021-06-28	133.210424	135.047667	133.150516	134.578369	62111300
524	2021-06-29	134.598343	136.285817	134.149019	136.126053	64556100
525	2021-06-30	135.966285	137.204435	135.666731	136.755112	63261400

526 rows × 6 columns

In [64]:
```
data.shape
```

Out[64]: (526, 6)

In [65]:
```
null_values=np.sum(data.isnull())

null_values
```

Out[65]:
```
Date      0
Open      0
High      0
Low       0
Close     0
Volume    0
dtype: int64
```

In [67]:
```
dup_value=np.sum(data.duplicated())
dup_value

```

Out[67]: 0

In [68]:
```
1 data.describe()
```

Out[68]:

	Open	High	Low	Close	Volume
count	526.000000	526.000000	526.000000	526.000000	5.260000e+02
mean	91.608297	92.723937	90.495177	91.658347	1.280820e+08
std	30.775590	31.076841	30.278098	30.670976	6.144065e+07
min	42.988224	43.595909	41.721424	42.463867	4.544800e+07
25%	63.748610	64.914196	63.372911	64.275314	8.736720e+07
50%	86.864926	87.608795	85.528444	86.946747	1.112724e+08
75%	122.473159	123.951533	120.681261	122.311106	1.507170e+08
max	142.928469	144.411483	140.708887	142.490524	4.265100e+08

In [69]:
```
1 data = pd.read_csv('AAPL.csv')
2 data=data.iloc[:,[0,1,2,3,4,5]]
3 data.head()
4
5
```

Out[69]:

	Date	Open	High	Low	Close	Volume
0	2019-05-31	43.181805	43.613062	42.877969	42.897572	108174400
1	2019-06-03	43.027439	43.595909	41.721424	42.463867	161584400
2	2019-06-04	42.988224	44.063910	42.762796	44.017353	123872000
3	2019-06-05	45.154295	45.328268	44.384898	44.727940	119093600
4	2019-06-06	44.860258	45.445882	44.632378	45.384624	90105200

In [70]:
```
1 # Convert 'Date' to numerical format
2 data['Date'] = pd.to_datetime(data['Date'])
3 data['Date_Ordinal'] = data['Date'].apply(lambda x: x.toordinal())
4
5 data.head()
```

Out[70]:

	Date	Open	High	Low	Close	Volume	Date_Ordinal
0	2019-05-31	43.181805	43.613062	42.877969	42.897572	108174400	737210
1	2019-06-03	43.027439	43.595909	41.721424	42.463867	161584400	737213
2	2019-06-04	42.988224	44.063910	42.762796	44.017353	123872000	737214
3	2019-06-05	45.154295	45.328268	44.384898	44.727940	119093600	737215
4	2019-06-06	44.860258	45.445882	44.632378	45.384624	90105200	737216

In [71]:
```
def polynomial_features(x, degree):

    return np.column_stack([x**i for i in range(degree + 1)])

X = data['Date_Ordinal'].values
y = data['Close'].values


degree = 2
X_poly = polynomial_features(X, degree)

print(X_poly)
```

```
[[           1       737210 543478584100]
 [           1       737213 543483007369]
 [           1       737214 543484481796]
 ...
 [           1       737969 544598244961]
 [           1       737970 544599720900]
 [           1       737971 544601196841]]
```

In [72]:
```
X_poly_inv = np.linalg.pinv(X_poly)
#coefficients
coefficients = X_poly_inv @ y

print('Coefficients:', coefficients)

```

```
Coefficients: [-3.59914065e-07 -1.32734617e-01  1.80125533e-07]
```

In [73]:

```
def predict_polynomial(X_range, coefficients):
    """Predict using the polynomial model."""
    X_poly = polynomial_features(X_range, len(coefficients) - 1)
    return X_poly @ coefficients

# Predict stock prices
X_range = np.linspace(X.min(), X.max(), 100)
y_pred = predict_polynomial(X_range, coefficients)

print("The new time that predict the close stock of the day")
pd.DataFrame(X_range)
```

```
The new time that predict the close stock of the day
```

Out[73]:

	0
0	737210.000000
1	737217.686869
2	737225.373737
3	737233.060606
4	737240.747475
...	...
95	737940.252525
96	737947.939394
97	737955.626263
98	737963.313131
99	737971.000000

100 rows × 1 columns

In [74]:
```
print("future close stock of the day")
pd.DataFrame(y_pred)
```

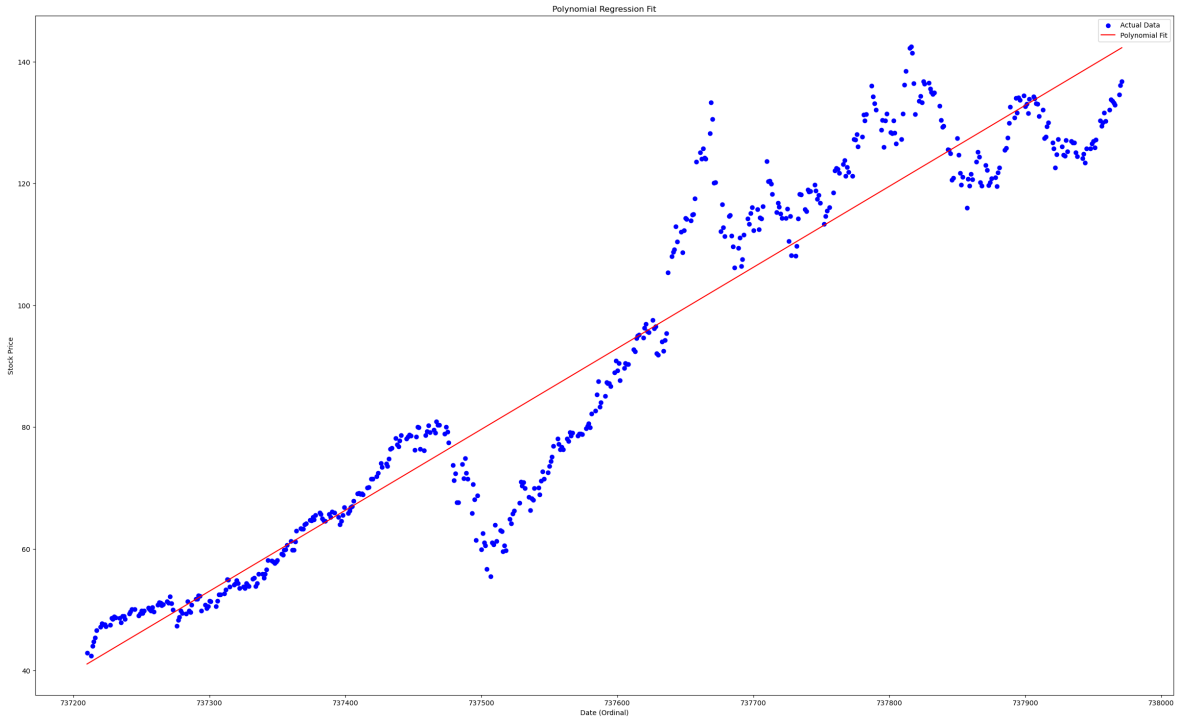
```
future close stock of the day
```

Out[74]:

	0
0	41.082424
1	42.103605
2	43.124807
3	44.146030
4	45.167275
...	...
95	138.189658
96	139.212861
97	140.236085
98	141.259331
99	142.282598

100 rows × 1 columns

In [75]:
```
# Plot the results
plt.figure(figsize=(30, 18))
plt.scatter(data['Date_Ordinal'], data['Close'], color='blue', label='Act
plt.plot(X_range, y_pred, color='red', label='Polynomial Fit')
plt.xlabel('Date (Ordinal)')
plt.ylabel('Stock Price')
plt.title('Polynomial Regression Fit')
plt.legend()
plt.show()
```



In [ ]:
```
1
```