In [27]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#%matplotlib inline enables the drawing of matplotlib figures in the IPython environment

import chart_studio.plotly as py
import plotly.graph_objs as go
from plotly.offline import plot

#for offline plotting
from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
```

In [28]:

```
# Reading input data set
dataset = pd.read_csv('G:\Dattathreya\PROJECT\DATA SETS\INR=X.csv')
dataset.head()
# returns top 5 rows from the data set
```

Out[28]:

	Date	Open	High	Low	Close	Adj Close	Volume
0	2020-01-27	71.320000	71.635002	71.320000	71.324997	71.324997	0
1	2020-01-28	71.654999	71.654999	71.178001	71.440002	71.440002	0
2	2020-01-29	71.230103	71.425003	71.168503	71.230400	71.230400	0
3	2020-01-30	71.300003	71.711998	71.300003	71.300003	71.300003	0
4	2020-01-31	71.639999	71.639999	71.277496	71.639999	71.639999	0

In [29]:

```
dataset.info()
# to find no.of colums in the dataset and also returns null values
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 262 entries, 0 to 261
Data columns (total 7 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   Date       262 non-null    object 
 1   Open       262 non-null    float64
 2   High       262 non-null    float64
 3   Low        262 non-null    float64
 4   Close      262 non-null    float64
 5   Adj Close  262 non-null    float64
 6   Volume     262 non-null    int64  
dtypes: float64(5), int64(1), object(1)
memory usage: 14.5+ KB
```

In [30]:

```
# changing date to date time function using pandas and datetime()
dataset['Date'] = pd.to_datetime(dataset['Date'])

# To print the time period of data and number of days.
# using formatted string litreal (f' ')
print(
    f'Dataframe contains stock prices between {dataset.Date.min()} {dataset.Date.max()}')
print(
    f'Total no.of days of data collected = {(dataset.Date.max() - dataset.Date.min()).days} days')
```

```
Dataframe contains stock prices between 2020-01-27 00:00:00 2021-01-26 00:00:00
Total no.of days of data collected = 365 days
```

In [31]:

```
# describe function gives the minimum, maximum, mean, standard deviation, and quatiles of the data
dataset.describe()
```

Out[31]:

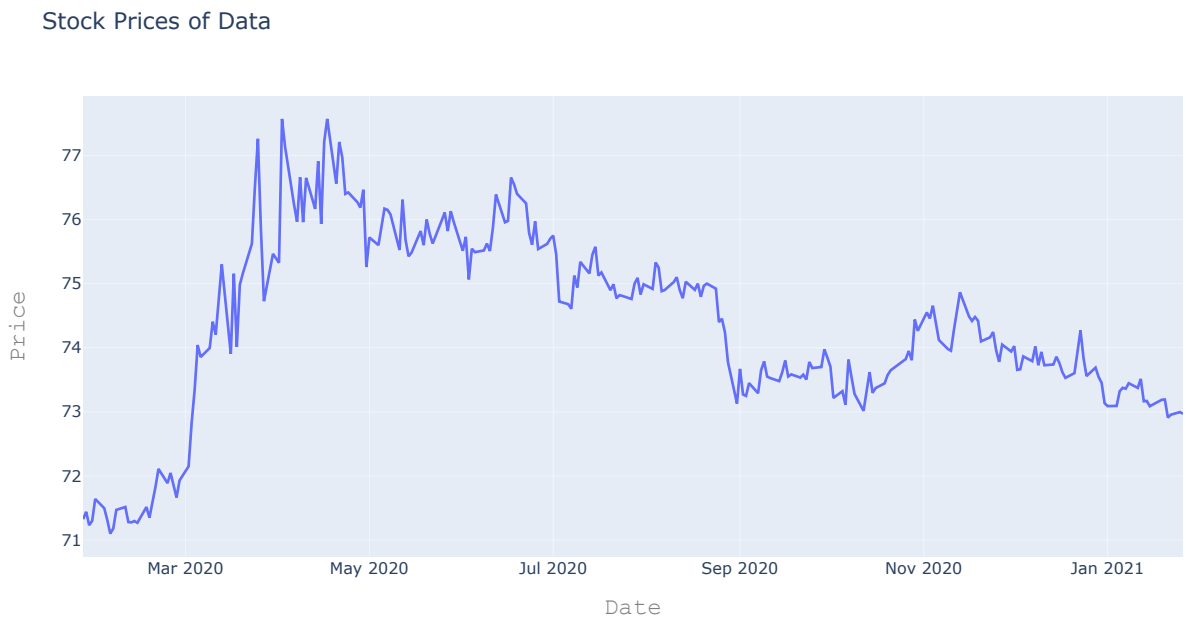
	Open	High	Low	Close	Adj Close	Volume
count	262.000000	262.000000	262.000000	262.000000	262.000000	262.0
mean	74.373533	74.631087	74.011048	74.358489	74.358489	0.0
std	1.417620	1.494521	1.311330	1.426245	1.426245	0.0
min	71.100403	71.279999	71.064003	71.099998	71.099998	0.0
25%	73.546175	73.706577	73.202003	73.531049	73.531049	0.0
50%	74.332001	74.531300	73.881748	74.275799	74.275799	0.0
75%	75.484551	75.737499	75.067053	75.489424	75.489424	0.0
max	77.684998	77.754997	76.496300	77.570000	77.570000	0.0

In [32]:

```
# Arranging the layout for our plot(graph)
layout = go.Layout(
    title='Stock Prices of Data',
    xaxis=dict(
        title='Date',
        titlefont=dict(
            family='Courier New, monospace',
            size=18,
            color='#7f7f7f'
        )
    ),
    yaxis=dict(
        title='Price',
        titlefont=dict(
            family='Courier New, monospace',
            size=18,
            color='#7f7f7f'
        )
    )
)

dataset_data = [{'x': dataset['Date'], 'y':dataset['Close']}]
plot = go.Figure(data=dataset_data, layout=layout)

# ploting the graph using iplot() function
# plot(iplot)
# plotting offline
iplot(plot)
```



In [33]:

```
# Building the regression model
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

#For preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

#for model evaluation and finding accuracy
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score

# Split the data into train and test sets
X = np.array(dataset.index).reshape(-1, 1)
Y = dataset['Close']
X_train, X_test, Y_train, Y_test = train_test_split(
    X, Y, test_size=0.3, random_state=101)

# Creating a linear model
lrm = LinearRegression()
lrm.fit(X_train, Y_train)
```

Out[33]:

```
LinearRegression()
```

In [34]:

```
# Plot actual and predicted values for train dataset
trainact = go.Scatter(
    x=X_train.T[0],
    y=Y_train,
    mode='markers',
    name='Actual'
)
trainpred = go.Scatter(
    x=X_train.T[0],
    y=lrm.predict(X_train).T,
    mode='lines',
    name='Predicted'
)
dataset_data = [trainact, trainpred]
layout.xaxis.title.text = 'Day'
plot2 = go.Figure(data=dataset_data, layout=layout)

iplot(plot2)
```

In [35]:

```
#Calculate scores for model evaluation
scores = f'''
{'Metric'.ljust(10)}{'Train'.center(20)}{'Test'.center(20)}
{'r2_score'.ljust(10)}{r2_score(Y_train, lrm.predict(X_train))}\t{r2_score(Y_test, lrm.predict(X_test))}
{'MSE'.ljust(10)}{mse(Y_train, lrm.predict(X_train))}\t{mse(Y_test, lrm.predict(X_test))}
'''
print(scores)
```

```
Metric           Train                Test
r2_score  0.01380593855830714   0.012899172316570895
MSE       2.120777510280475     1.692810301949111
```