In [2]:
```
import pandas as pd
import numpy as np
from sklearn import metrics
%matplotlib inline
import matplotlib.pyplot as plt
```

In [3]:
```
dataset=pd.read_csv(r"C:\Users\SWAJAN\Documents\education\da project\datasets\tcs_stock.csv")
```

In [5]:
```
dataset.head()
```

Out[5]:

	Date	Symbol	Series	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades	Deliverable Volume	%Deliverble
0	2015-01-01	TCS	EQ	2558.25	2567.0	2567.00	2541.00	2550.00	2545.55	2548.51	183415	4.674345e+13	8002	52870	0.2883
1	2015-01-02	TCS	EQ	2545.55	2551.0	2590.95	2550.60	2588.40	2579.45	2568.19	462870	1.188740e+14	27585	309350	0.6683
2	2015-01-05	TCS	EQ	2579.45	2581.0	2599.90	2524.65	2538.10	2540.25	2563.94	877121	2.248886e+14	43234	456728	0.5207
3	2015-01-06	TCS	EQ	2540.25	2529.1	2529.10	2440.00	2450.05	2446.60	2466.90	1211892	2.989615e+14	84503	714306	0.5894
4	2015-01-07	TCS	EQ	2446.60	2470.0	2479.15	2407.45	2426.90	2417.70	2433.96	1318166	3.208362e+14	101741	886368	0.6724

In [6]:
```
dataset.shape
```

Out[6]:
```
(248, 15)
```

In [5]:
```
dataset.isnull().sum()
```

Out[5]:
```
Date                 0
Symbol               0
Series               0
Prev Close           0
Open                 0
High                 0
Low                  0
Last                 0
Close                0
VWAP                 0
Volume               0
Turnover             0
Trades               0
Deliverable Volume   0
%Deliverble          0
dtype: int64
```

In [7]:
```
dataset.isna().any()
```

Out[7]:
```
Date                 False
Symbol               False
Series               False
Prev Close           False
Open                 False
High                 False
Low                  False
Last                 False
Close                False
VWAP                 False
Volume               False
Turnover             False
Trades               False
Deliverable Volume   False
%Deliverble          False
dtype: bool
```

In [8]:
```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 248 entries, 0 to 247
Data columns (total 15 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Date                248 non-null    object 
 1   Symbol              248 non-null    object 
 2   Series              248 non-null    object 
 3   Prev Close          248 non-null    float64
 4   Open                248 non-null    float64
 5   High                248 non-null    float64
 6   Low                 248 non-null    float64
 7   Last                248 non-null    float64
 8   Close               248 non-null    float64
 9   VWAP                248 non-null    float64
 10  Volume              248 non-null    int64  
 11  Turnover            248 non-null    float64
 12  Trades              248 non-null    int64  
 13  Deliverable Volume  248 non-null    int64  
 14  %Deliverble         248 non-null    float64
dtypes: float64(9), int64(3), object(3)
memory usage: 29.2+ KB
```

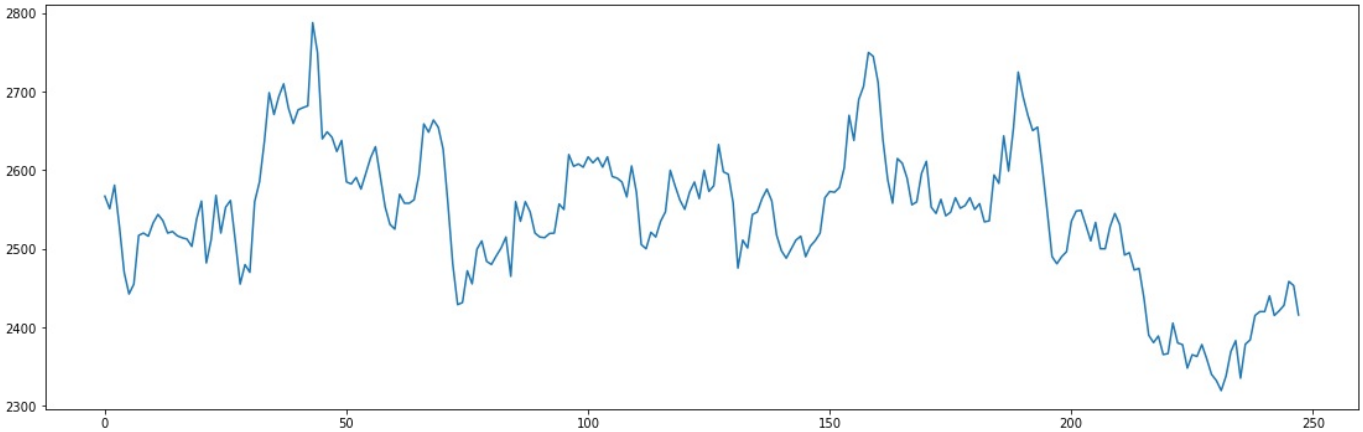
In [9]:
```
dataset.describe()
```

Out[9]:

	Prev Close	Open	High	Low	Last	Close	VWAP	Volume	Turnover	Trades
count	248.000000	248.000000	248.000000	248.000000	248.000000	248.000000	248.000000	2.480000e+02	2.480000e+02	248.000000
mean	2538.207460	2542.172782	2563.580444	2514.408468	2538.039718	2537.717944	2538.432137	1.172296e+06	2.977489e+14	66873.608871
std	86.829359	87.605699	90.598368	82.952778	86.849305	87.057814	86.813053	6.220635e+05	1.576443e+14	28882.906787
min	2319.800000	2319.400000	2343.900000	2315.250000	2321.000000	2319.800000	2322.270000	6.758200e+04	1.667550e+13	5197.000000
25%	2495.312500	2499.500000	2518.900000	2472.100000	2497.500000	2495.150000	2496.665000	7.821352e+05	1.950716e+14	45476.250000
50%	2543.050000	2548.500000	2566.000000	2520.000000	2540.150000	2541.475000	2540.445000	1.031024e+06	2.631783e+14	61449.500000
75%	2592.000000	2594.250000	2615.750000	2567.300000	2593.425000	2592.000000	2592.607500	1.393266e+06	3.550390e+14	82066.750000
max	2776.000000	2788.000000	2812.100000	2721.900000	2785.100000	2776.000000	2763.040000	4.834371e+06	1.206435e+15	211247.000000

In [10]:
```
dataset['Open'].plot(figsize=(19,6))
```

Out[10]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bcdb466580>
```



In [11]:
```
import seaborn as sns
```

In [12]:
```
x=dataset[['Open','High','Low','Volume']]
y=dataset['Close']
```

In [13]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train , y_test= train_test_split(x,y,test_size=0.2,random_state=0)
```

In [14]:
```
x_train.shape
x_test.shape
```

```
(50, 4)
```

Out[14]: (50, 4)

In [15]:
```
x_train.shape
```

Out[15]: (198, 4)

In [16]:
```
from sklearn.linear_model import LinearRegression
```

In [17]:
```
 from sklearn.metrics import accuracy_score
```

In [18]:
```
reg=LinearRegression()
```

In [19]:
```
reg.fit(x_train,y_train)
```

Out[19]: LinearRegression()

In [20]:
```
reg.coef_
```

Out[20]: array([-4.81179028e-01, 6.89821364e-01, 7.90419813e-01, -1.71386863e-06])

In [21]:
```
reg.intercept_
```

Out[21]: 7.124588444366054

In [22]:
```
predicted=reg.predict(x_test)
```

In [23]:
```
predicted.shape
```

Out[23]: (50,)

In [24]:
```
dframe=pd.DataFrame( y_test, predicted )
```

In [25]:
```
dfr=pd.DataFrame( { 'Actual Price':y_test, 'Predicted price': predicted } )
```

In [26]:
```
dfr.head()
```

Out[26]:

	Actual Price	Predicted price
247	2436.85	2435.616791
168	2591.80	2593.855536
76	2493.05	2491.382122
150	2552.05	2556.142112
145	2510.75	2510.788191

In [27]:
```
reg.score(x_test,y_test)
```

Out[27]: 0.9921391734686681

In [28]:
```
import math

print('mean absolute error : ', metrics.mean_absolute_error(y_test,predicted))
```

```
print('mean squared error : ', metrics.mean_squared_error(y_test,predicted))

print('root mean squared error : ',math.sqrt(metrics.mean_squared_error(y_test,predicted)))
```

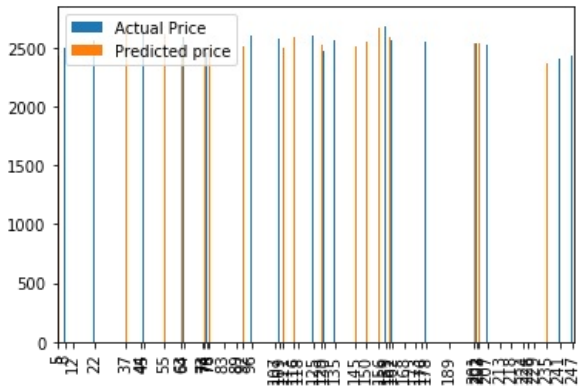
```
mean absolute error :  6.101846635684333
mean squared error :  64.0330627048773
root mean squared error :  8.002066152243263
```

In [29]:

```
graph=dfr.head(50)
graph.plot(kind="bar")
```

Out[29]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1bcdda43ca0>
```

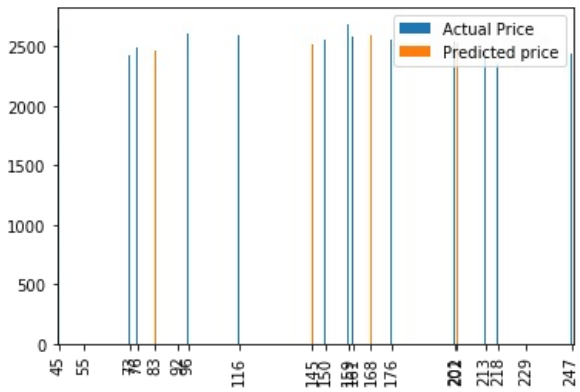


In [30]:

```
graph=dfr.head(20)
graph.plot(kind="bar")
```

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1bcdda61940>
```



In [32]:

```
dfr.head(50)
```

Out[32]:

	Actual Price	Predicted price
247	2436.85	2435.616791
168	2591.80	2593.855536
76	2493.05	2491.382122
150	2552.05	2556.142112
145	2510.75	2510.788191
73	2427.05	2425.495139
45	2646.80	2645.920268
159	2686.85	2689.836348
218	2351.45	2358.572305
213	2458.15	2461.241897
96	2601.00	2608.371654
201	2537.15	2534.371972
83	2463.05	2470.064258
176	2551.65	2550.967825

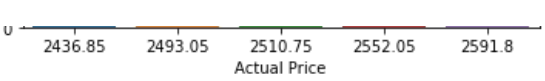
161	2578.65	2588.517153
202	2536.55	2525.726531
55	2617.15	2621.771813
116	2591.50	2587.352072
229	2328.40	2333.958960
92	2512.70	2511.893226
203	2531.10	2536.067085
135	2564.40	2563.925068
162	2567.15	2571.338282
89	2499.25	2495.974742
44	2696.45	2712.462797
207	2517.35	2509.536614
37	2672.20	2688.029620
111	2504.80	2499.154673
63	2547.30	2518.996247
109	2571.30	2575.238667
118	2570.30	2572.434320
8	2497.90	2499.139247
189	2700.00	2694.441307
64	2585.00	2576.184150
129	2522.50	2528.727911
5	2443.80	2433.174678
22	2558.25	2547.615038
125	2603.90	2600.013940
12	2511.00	2524.036982
173	2549.75	2558.562499
241	2405.05	2411.881888
226	2361.90	2372.159743
107	2561.65	2564.065118
156	2684.75	2670.916782
75	2457.25	2446.611253
178	2550.35	2554.094761
235	2378.45	2369.901732
130	2474.35	2474.439559
74	2445.10	2442.453414
224	2353.35	2347.080791

In [33]:
```
reg.score(x_test,y_test)
```

Out[33]:
```
0.9921391734686681
```

In [38]:
```
import seaborn as sns
sns.barplot(x="Actual Price",y="Predicted price",data=dfr.head(5))
```

Out[38]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1bce016c880>
```



In [ ]:

In [ ]: