In [3]:
```
import pandas as pd
import numpy as np
from sklearn import metrics
%matplotlib inline
import matplotlib.pyplot as plt
```

In [4]:
```
dataset=pd.read_csv(r"C:\Users\SWAJAN\Documents\education\da project\datasets\infy_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	INFY	EQ	1972.55	1968.95	1982.00	1956.9	1971.00	1974.40	1971.34	500691	9.870306e+13	14908	258080	0.5154
1	2015-01-02	INFY	EQ	1974.40	1972.00	2019.05	1972.0	2017.95	2013.20	2003.25	1694580	3.394669e+14	54166	1249104	0.737
2	2015-01-05	INFY	EQ	2013.20	2009.90	2030.00	1977.5	1996.00	1995.90	2004.59	2484256	4.979911e+14	82694	1830962	0.737
3	2015-01-06	INFY	EQ	1995.90	1980.00	1985.00	1934.1	1965.10	1954.20	1954.82	2416829	4.724458e+14	108209	1772070	0.733
4	2015-01-07	INFY	EQ	1954.20	1965.00	1974.75	1950.0	1966.05	1963.55	1962.59	1812479	3.557162e+14	62463	1317720	0.727

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

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

In [7]:
```
dataset.isnull().sum()
```

Out[7]:
```
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 [8]:
```
dataset.isna().any()
```

Out[8]:
```
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 [9]:
```
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
```

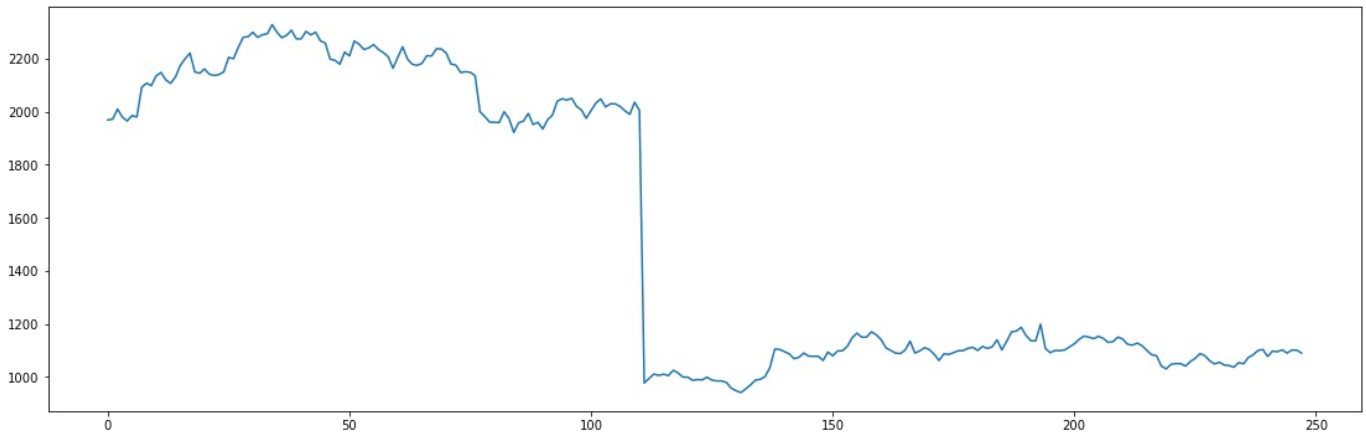
In [10]:
```
dataset.describe()
```

Out[10]:

	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	1551.474798	1550.506855	1566.266532	1530.085887	1548.084879	1547.978226	1548.133589	2.982072e+06	4.234132e+14	92675.024194
std	529.396894	530.578342	534.714088	524.194873	529.493276	529.468189	528.861589	2.043627e+06	2.708337e+14	50541.614178
min	937.500000	941.000000	952.100000	932.650000	935.500000	937.500000	941.180000	3.536520e+05	3.923481e+13	13196.000000
25%	1085.912500	1088.000000	1099.975000	1067.150000	1086.875000	1085.912500	1085.907500	1.722753e+06	2.847065e+14	63052.250000
50%	1149.650000	1150.000000	1159.725000	1131.150000	1145.625000	1149.325000	1146.245000	2.532474e+06	3.624709e+14	80019.000000
75%	2125.312500	2136.137500	2150.000000	2104.500000	2125.250000	2125.312500	2125.082500	3.567063e+06	4.915434e+14	106617.250000
max	2324.700000	2328.500000	2336.000000	2292.050000	2323.200000	2324.700000	2322.170000	1.915506e+07	2.285439e+15	408583.000000

In [11]:
```
dataset['Open'].plot(figsize=(19,6))
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1b60e6a6370>
```



In [12]:
```
import seaborn as sns
```

In [13]:
```
x=dataset[['Open','High','Low','Volume']]
y=dataset['Close']
```

In [14]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train , y_test= train_test_split(x,y,test_size=0.1,random_state=0)
```

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

```
(25, 4)
```

Out[15]: (25, 4)

In [16]:
```
x_train.shape
```

Out[16]: (223, 4)

In [17]:
```
from sklearn.linear_model import LinearRegression
```

In [18]:
```
 from sklearn.metrics import accuracy_score
```

In [19]:
```
reg=LinearRegression()
```

In [20]:
```
reg.fit(x_train,y_train)
```

Out[20]: LinearRegression()

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

Out[21]: array([-5.13528790e-01, 8.80773340e-01, 6.31218854e-01, -7.74699148e-07])

In [22]:
```
reg.intercept_
```

Out[22]: 1.4975643538564327

In [23]:
```
predicted=reg.predict(x_test)
```

In [24]:
```
predicted.shape
```

Out[24]: (25,)

In [25]:
```
dframe=pd.DataFrame( y_test, predicted )
```

In [26]:
```
dfr=pd.DataFrame( { 'Actual Price':y_test, 'Predicted price': predicted } )
```

In [27]:
```
dfr.head()
```

Out[27]:

	Actual Price	Predicted price
247	1105.40	1102.346336
168	1099.45	1103.406374
76	1995.20	2039.866058
150	1095.60	1095.712094
145	1077.05	1066.075250

In [28]:
```
reg.score(x_test,y_test)
```

Out[28]: 0.9992825576971539

In [29]:
```
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)))
```

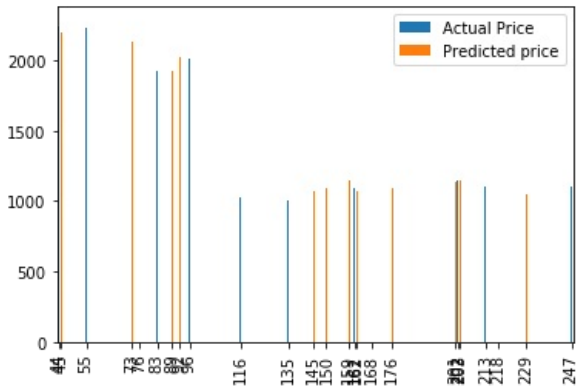
```
mean absolute error :  9.108103880953577
mean squared error :  166.08161449578975
root mean squared error :  12.88726559421314
```

In [30]:

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

Out[30]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b610c4f100>
```

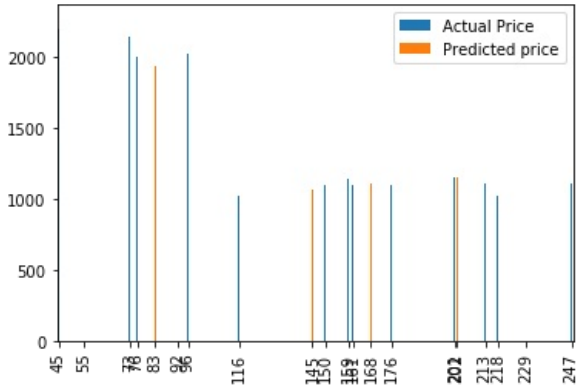


In [31]:

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

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1b6108cc820>
```



In [32]:

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

Out[32]:

	Actual Price	Predicted price
247	1105.40	1102.346336
168	1099.45	1103.406374
76	1995.20	2039.866058
150	1095.60	1095.712094
145	1077.05	1066.075250
73	2142.60	2133.119289
45	2190.05	2204.982446
159	1134.55	1144.274439
218	1020.00	1021.852404
213	1103.85	1108.156839
96	2016.80	2025.843407
201	1149.90	1138.529365
83	1922.05	1933.743807
176	1099.75	1090.352403

161	1092.05	1095.976985
202	1152.15	1149.601141
55	2233.45	2252.070492
116	1023.85	1018.661152
229	1049.05	1054.027865
92	2022.35	2025.563290
203	1149.40	1148.371244
135	1001.85	1000.330349
162	1085.65	1076.189079
89	1934.80	1925.159735
44	2247.60	2270.611438

In [37]:
```
reg.score(x_test,y_test)
reg.score(x_train, y_train)
```

Out[37]:
```
0.9996967320810379
```

In [40]:
```
from sklearn.metrics import r2_score
r2_score(y_test,predicted)
```

Out[40]:
```
0.9992825576971539
```

In [ ]:
```

```