**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

**from** **sklearn.model\_selection** **import** train\_test\_split

**from** **sklearn.linear\_model** **import** LinearRegression

**from** **sklearn** **import** metrics

In [3]:

tesla\_data=pd.read\_csv("TSLA.csv")

In [31]:

print(tesla\_data)

Date Open High Low Close Adj Close \

0 2010-06-30 5.158000 6.084000 4.660000 4.766000 4.766000

1 2010-07-01 5.000000 5.184000 4.054000 4.392000 4.392000

2 2010-07-02 4.600000 4.620000 3.742000 3.840000 3.840000

3 2010-07-06 4.000000 4.000000 3.166000 3.222000 3.222000

4 2010-07-07 3.280000 3.326000 2.996000 3.160000 3.160000

... ... ... ... ... ... ...

2574 2020-09-21 453.130005 455.679993 407.070007 449.390015 449.390015

2575 2020-09-22 429.600006 437.760010 417.600006 424.230011 424.230011

2576 2020-09-23 405.160004 412.149994 375.880005 380.359985 380.359985

2577 2020-09-24 363.799988 399.500000 351.299988 387.790009 387.790009

2578 2020-09-25 393.470001 408.730011 391.299988 407.339996 407.339996

Volume

0 85935500

1 41094000

2 25699000

3 34334500

4 34608500

... ...

2574 109476800

2575 79580800

2576 95074200

2577 96561100

2578 67068400

[2579 rows x 7 columns]

In [33]:

tesla\_data.info()

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

RangeIndex: 2579 entries, 0 to 2578

Data columns (total 7 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Date 2579 non-null object

1 Open 2579 non-null float64

2 High 2579 non-null float64

3 Low 2579 non-null float64

4 Close 2579 non-null float64

5 Adj Close 2579 non-null float64

6 Volume 2579 non-null int64

dtypes: float64(5), int64(1), object(1)

memory usage: 141.2+ KB

In [5]:

tesla\_data.head()

Out[5]:

|  | **Date** | **Open** | **High** | **Low** | **Close** | **Adj Close** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 2010-06-30 | 5.158 | 6.084 | 4.660 | 4.766 | 4.766 | 85935500 |
| **1** | 2010-07-01 | 5.000 | 5.184 | 4.054 | 4.392 | 4.392 | 41094000 |
| **2** | 2010-07-02 | 4.600 | 4.620 | 3.742 | 3.840 | 3.840 | 25699000 |
| **3** | 2010-07-06 | 4.000 | 4.000 | 3.166 | 3.222 | 3.222 | 34334500 |
| **4** | 2010-07-07 | 3.280 | 3.326 | 2.996 | 3.160 | 3.160 | 34608500 |

In [12]:

tesla\_data.describe()

Out[12]:

|  | **Open** | **High** | **Low** | **Close** | **Adj Close** | **Volume** |
| --- | --- | --- | --- | --- | --- | --- |
| **count** | 2579.000000 | 2579.000000 | 2579.000000 | 2579.000000 | 2579.000000 | 2.579000e+03 |
| **mean** | 49.206686 | 50.301806 | 48.073117 | 49.253279 | 49.253279 | 3.078217e+07 |
| **std** | 57.934102 | 59.888383 | 55.852349 | 58.119783 | 58.119783 | 2.855717e+07 |
| **min** | 3.228000 | 3.326000 | 2.996000 | 3.160000 | 3.160000 | 5.925000e+05 |
| **25%** | 7.159000 | 7.268000 | 6.989000 | 7.153000 | 7.153000 | 1.047400e+07 |
| **50%** | 44.001999 | 44.660000 | 43.301998 | 43.924000 | 43.924000 | 2.413100e+07 |
| **75%** | 59.339000 | 60.171000 | 57.841000 | 59.020000 | 59.020000 | 3.979150e+07 |
| **max** | 502.140015 | 502.489990 | 470.510010 | 498.320007 | 498.320007 | 3.046940e+08 |

In [13]:

tesla\_data.shape

Out[13]:

(2579, 7)

In [19]:

tesla\_data.isnull().sum()

Out[19]:

Date 0

Open 0

High 0

Low 0

Close 0

Adj Close 0

Volume 0

dtype: int64

In [38]:

x = tesla\_data[['High', 'Low', 'Open', 'Volume']].values

y = tesla\_data['Close'].values

In [53]:

print(x)

print(y)

[[6.08400000e+00 4.66000000e+00 5.15800000e+00 8.59355000e+07]

[5.18400000e+00 4.05400000e+00 5.00000000e+00 4.10940000e+07]

[4.62000000e+00 3.74200000e+00 4.60000000e+00 2.56990000e+07]

...

[4.12149994e+02 3.75880005e+02 4.05160004e+02 9.50742000e+07]

[3.99500000e+02 3.51299988e+02 3.63799988e+02 9.65611000e+07]

[4.08730011e+02 3.91299988e+02 3.93470001e+02 6.70684000e+07]]

[ 4.766 4.392 3.84 ... 380.359985 387.790009 407.339996]

In [39]:

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y, test\_size=0.2,random\_state=1)

In [40]:

regressor = LinearRegression()

In [41]:

regressor.fit(x\_train, y\_train)

Out[41]:

LinearRegression()

In [42]:

print(regressor.coef\_)

[ 8.86184318e-01 5.97055117e-01 -4.86156079e-01 -8.50537074e-09]

In [43]:

print(regressor.intercept\_)

0.16107831311390441

In [44]:

predicted = regressor.predict(x\_test)

In [45]:

print(predicted)

[ 48.31506237 5.42628358 67.82052749 6.65998696 50.582403

50.1576105 41.87815496 24.44502951 49.58917194 43.88499588

298.9478631 37.55380299 7.8271123 43.76849087 54.55689971

4.81946491 43.08818331 62.27772735 6.57705705 50.89518687

162.31673838 54.07935092 26.92665852 17.45858966 67.53368876

45.30204561 72.86086135 49.07391744 143.36788222 51.92440099

50.6970579 29.53466922 49.57999154 47.45795379 32.82006525

155.14415191 62.71290218 4.31383714 43.68572232 27.59572087

58.71308847 38.2490949 39.05405015 38.5829505 23.66716596

39.00697909 3.42828656 40.97496353 7.96752939 52.47220087

3.9260151 5.69849224 3.00835442 42.26695342 33.1761977

7.26818166 7.16881571 49.52592057 449.84776532 8.23641207

130.90476805 47.11014685 70.12248134 41.81977789 43.79932994

55.30901315 126.83777272 64.02847146 48.66145103 67.95410459

5.09695858 58.17475452 39.74929918 41.26404001 6.75883911

37.32206012 34.8954167 5.63612744 53.95951624 45.23545598

5.55307457 72.55841475 54.89438645 38.41095059 5.73727865

201.61393487 10.47223681 4.19621908 43.81599745 39.73686454

45.0024549 5.83773005 4.14174857 4.9819291 65.06303934

68.19184866 53.39442245 4.77641581 45.57883286 6.04904104

51.25889758 6.96499379 46.50851518 71.4155589 308.5751515

44.62976925 61.10269501 43.32226825 5.94935628 4.88090147

30.71675842 5.61260373 50.08470735 63.99183834 7.06740902

300.55719744 30.3583319 24.80577031 50.22493569 6.25965551

46.6549413 7.02480441 70.86518845 50.53236185 46.29530312

63.65619941 5.68163138 45.88268966 68.93621711 34.99089394

54.36321181 45.60195064 46.85621299 57.40005864 69.35482375

39.6770602 52.35614187 28.93555835 7.45301572 45.75298293

68.38370151 5.21777997 10.85273612 39.03647114 43.78996132

46.46955869 37.01345665 6.33677966 43.45478235 44.00213899

51.33064135 50.1988908 85.17379677 66.0373643 44.09544461

65.68058897 61.21168216 53.66238167 35.72689327 37.91137625

50.56052434 51.65887619 61.54027003 6.50046732 4.22684896

5.54132649 53.18381611 35.88477455 59.79190199 5.84108335

48.74834818 32.94989247 69.89417505 7.1415091 24.21311176

94.84592758 4.94544788 44.08908141 4.98240483 37.94173508

41.72025578 5.66437854 73.54843224 6.18080667 62.73492272

42.37243606 11.21244877 46.7593647 55.471276 40.30381736

6.45148394 5.10216822 6.07296692 5.96391728 67.61997579

157.02904279 29.76984808 67.92612947 41.26387233 49.73933498

34.74632469 76.16149199 42.29365269 39.39729045 42.15529288

25.37416158 20.43737951 60.95104588 42.35575513 5.44134516

45.10171419 33.58933179 46.39713281 42.41057597 38.66309924

24.10936449 30.6643811 44.3716604 45.43007115 6.02967688

37.48281302 6.57374162 7.64426197 96.95703846 45.4331266

4.26602525 5.72775161 61.84652532 300.27543544 274.83822148

50.88550511 6.87261747 6.75535716 56.42709079 6.01567623

59.90951835 5.81454775 36.37645418 149.02559291 71.85290954

56.03321942 35.89905594 46.18943911 7.2871897 71.20025848

64.36544909 49.58924538 47.50650301 44.38785239 56.56000249

6.16082621 58.87105158 42.40754223 42.3851499 33.85356753

40.51032043 44.98836763 6.74414673 4.88868078 50.02857065

4.73051303 63.68817331 24.69202581 45.55339356 58.27252138

50.06791969 6.16625275 40.96931316 32.60189106 40.27158889

48.21779083 37.79313732 44.2591156 43.93863939 6.33644089

5.49848935 5.38581305 7.09850003 4.16836429 42.36660192

6.91413762 54.39792793 40.97458612 54.43168315 66.15166695

69.02160718 6.67883153 68.29770454 50.99677782 47.80809878

4.30234816 6.27030193 57.70102627 24.28778754 7.25681103

59.62491137 6.62720698 65.93878392 47.32882539 53.1433184

49.09778484 40.66947134 38.59857028 64.11292328 47.30626696

161.13358521 97.04081154 55.85537148 18.81792118 50.00405233

5.07099463 27.26559358 40.98863404 44.22609428 111.96151796

50.22197824 6.05775828 55.47537037 6.9170881 5.63850015

4.31669204 51.03424726 42.87547097 56.04331483 44.87087315

7.1108996 61.91630256 6.47150872 65.65576802 5.60117681

63.49195489 174.52611567 52.84398208 39.63331785 5.66568204

48.78953713 6.94106726 6.29025804 6.65228055 73.20773854

6.41614529 40.88463768 8.89874259 49.70185786 5.94709292

5.93455062 37.85921582 5.75005344 5.52056009 49.7530841

31.32932607 52.24157301 5.34999254 62.522181 7.50217816

55.56223749 6.15455296 5.95739438 46.94178283 6.73456476

44.72614011 69.81761806 6.8482292 7.94238505 36.62196025

5.61767128 5.52936279 61.96484184 66.71843889 44.42084656

48.79714359 68.39342985 44.08742389 47.31588505 42.02331206

59.20321412 6.7707586 51.67958924 51.20626938 38.63702623

46.19595542 53.91028428 39.39546908 162.90011993 64.36385776

5.89078708 45.49679611 38.94279206 61.34757337 59.01504739

67.77099358 8.96301074 69.84323853 6.96779699 44.54485899

51.89232876 65.99002649 163.40753742 5.09894892 427.89783613

49.70662798 39.49036326 62.53462266 37.72635746 69.7363672

114.11519672 134.13348539 155.05460823 392.04337991 6.64758355

58.11390738 43.15681113 5.69246449 46.39067285 71.79205592

45.56208948 199.76239054 50.61302124 39.37934213 62.00798416

67.58460424 52.87238723 6.08055298 35.3679325 41.995191

113.25767507 40.2849623 9.10956572 49.4109716 67.41674781

57.05889893 43.50985153 52.45913871 5.63695513 39.12785824

40.33524738 27.54269339 36.45444206 40.60606827 6.30267402

56.54230146 62.41708379 20.4188457 46.45252496 6.93295212

44.38343712 5.9887249 66.21292523 41.2822415 41.46234275

44.84823922 63.7342395 7.2927599 7.69736901 5.98869873

7.57963024 45.91735584 40.19749954 5.50940161 43.10343502

5.05761458 50.42606648 44.65264786 5.81514756 62.42204834

41.72788192 9.99685762 40.47753208 62.951237 57.23278336

66.86481341 6.36387325 62.70890335 5.67806338 68.19063362

68.99739333 63.77608715 363.15227374 44.94887609 52.43786384

105.70898239 58.20272383 83.31101456 23.26397466 39.6102804

39.94224454 69.10713884 63.20320504 4.48657503 49.24181204

6.17741393 46.90095972 47.48321516 46.49421756 5.44098983

7.85183491 5.23953491 5.65356548 5.86346134 68.69528904

61.14689246 70.45862797 4.56784923 6.02442216 6.04050281

322.45103567 45.47272733 60.00246367 41.12738825 4.24013449

67.23481387 5.40200602 6.95275534 7.74309833 5.83432863

74.10534747]

In [47]:

df2 = pd.DataFrame({'Actual': y\_test.flatten(), 'Predicted' : predicted.flatten()})

In [48]:

df2.head(25)

Out[48]:

|  | **Actual** | **Predicted** |
| --- | --- | --- |
| **0** | 48.598000 | 48.315062 |
| **1** | 5.348000 | 5.426284 |
| **2** | 68.570000 | 67.820527 |
| **3** | 6.430000 | 6.659987 |
| **4** | 49.812000 | 50.582403 |
| **5** | 50.004002 | 50.157610 |
| **6** | 41.400002 | 41.878155 |
| **7** | 24.690001 | 24.445030 |
| **8** | 49.785999 | 49.589172 |
| **9** | 43.472000 | 43.884996 |
| **10** | 297.000000 | 298.947863 |
| **11** | 38.782001 | 37.553803 |
| **12** | 7.708000 | 7.827112 |
| **13** | 43.888000 | 43.768491 |
| **14** | 53.790001 | 54.556900 |
| **15** | 4.650000 | 4.819465 |
| **16** | 43.529999 | 43.088183 |
| **17** | 62.924000 | 62.277727 |
| **18** | 6.426000 | 6.577057 |
| **19** | 51.400002 | 50.895187 |
| **20** | 163.884003 | 162.316738 |
| **21** | 54.992001 | 54.079351 |
| **22** | 27.090000 | 26.926659 |
| **23** | 18.450001 | 17.458590 |
| **24** | 66.185997 | 67.533689 |

In [49]:

**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: 0.4542042162062182

Mean Squared Error: 0.9482685456475671

Root Mean Squared Error: 0.9737908120574804

In [50]:

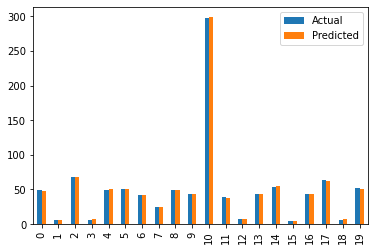
graph = df2.head(20)

In [51]:

graph.plot(kind='bar')

Out[51]:

<AxesSubplot:>



In [ ]:

In [ ]:

In [ ]:

In [ ]: