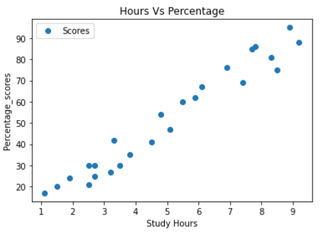
In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
url='https://bit.ly/w-data'
s_data=pd.read_csv(url)
s_data.head(10)
```

```
Matplotlib is building the font cache; this may take a moment.
```

Out[1]:

	Hours	Scores
0	2.5	21
1	5.1	47
2	3.2	27
3	8.5	75
4	3.5	30
5	1.5	20
6	9.2	88
7	5.5	60
8	8.3	81
9	2.7	25

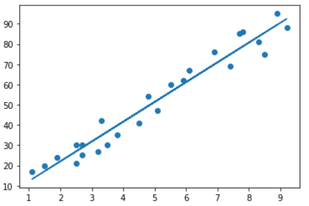
In [2]:
```
s_data.plot(x='Hours', y='Scores', style='o')
plt.title('Hours Vs Percentage')
plt.xlabel('Study Hours')
plt.ylabel('Percentage_scores')
plt.show()
```



In [11]:
```
x = s_data.iloc[:, :-1].values
y = s_data.iloc[:, 1].values
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)
from sklearn.linear_model import LinearRegression
regressor1 = LinearRegression()
regressor1.fit(x,y)
```

Out[11]:
```
LinearRegression()
```

In [12]:
```
line = regressor1.coef_*x+regressor1.intercept_
plt.scatter(x, y)
plt.plot(x, line);
plt.show()
```



In [13]:
```
print(x_test)
y_pred=regressor1.predict(x_test)
```

```
[[1.5]
 [3.2]
 [7.4]
 [2.5]
 [5.9]]
```

In [15]:
```
df=pd.DataFrame({'actual':y_test , 'predicted':y_pred})
df
```

Out[15]:

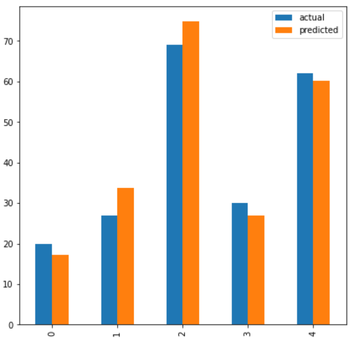
	actual	predicted
0	20	17.147378
1	27	33.766244
2	69	74.824618
3	30	26.923182
4	62	60.160913

In [16]:
```
print('training score:', regressor1.score(x_train,y_train))
print('testing score:', regressor1.score(x_test,y_test))
```

```
training score: 0.9512837351709387
testing score: 0.9491748734859171
```

In [17]:
```
df.plot(kind='bar', figsize=(7,7))
```

Out[17]:
```
<AxesSubplot:>
```



In [18]:
```
hours=9.25
test=np.array([hours])
test=test.reshape(-1,1)
own_pred=regressor1.predict(test)
print('no of hours={}'.format(hours))
print('predicted scores={}'.format(own_pred[0]))
```

```
no of hours=9.25
predicted scores=92.90985477015731
```

In [19]:
```
import numpy as np
from sklearn import metrics
print('mean absolute error:', metrics.mean_absolute_error(y_test,y_pred))
print('mean squared error:', metrics.mean_squared_error(y_test,y_pred))
print('root mean squared error:', np.sqrt(metrics.mean_squared_error(y_test,y_pred)))
print('explained variance score:', metrics.explained_variance_score(y_test,y_pred))
```

```
mean absolute error: 4.071877793635608
mean squared error: 20.1389481299402
root mean squared error: 4.487643939746134
explained variance score: 0.951522433518808
```

In [ ]:
```
```