In [24]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import math
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [25]:

```
df=pd.read_csv('BostonHousing.csv')
```

In [26]:

```
df.info()
# count This shows the number of non-null values in each numerical column
#mean
#std ,
#25%= Q1  value below which 25% of the data falls.
# 50% median
#75%= Q3   value below which 75% of the data falls.
#max and min val in that column
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64  
 4   nox      506 non-null    float64
 5   rm       501 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64  
 9   tax      506 non-null    int64  
 10  ptratio  506 non-null    float64
 11  b        506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

In [27]:

```
df.describe()
```

Out[27]:

|  | crim | zn | indus | chas | nox | rm |
|---|---|---|---|---|---|---|
| **count** | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 506.000000 | 501.000000 |
| **mean** | 3.613524 | 11.363636 | 11.136779 | 0.069170 | 0.554695 | 6.284341 |
| **std** | 8.601545 | 23.322453 | 6.860353 | 0.253994 | 0.115878 | 0.705587 |
| **min** | 0.006320 | 0.000000 | 0.460000 | 0.000000 | 0.385000 | 3.561000 |
| **25%** | 0.082045 | 0.000000 | 5.190000 | 0.000000 | 0.449000 | 5.884000 |
| **50%** | 0.256510 | 0.000000 | 9.690000 | 0.000000 | 0.538000 | 6.208000 |
| **75%** | 3.677083 | 12.500000 | 18.100000 | 0.000000 | 0.624000 | 6.625000 |
| **max** | 88.976200 | 100.000000 | 27.740000 | 1.000000 | 0.871000 | 8.780000 |

In [28]:

```
df.head()
```

Out[28]:

|  | crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | b |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.00632 | 18.0 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1 | 296 | 15.3 | 396.90 |
| **1** | 0.02731 | 0.0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 396.90 |
| **2** | 0.02729 | 0.0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 392.83 |
| **3** | 0.03237 | 0.0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 394.63 |
| **4** | 0.06905 | 0.0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 396.90 |

In [29]:

```
df.shape
```

Out[29]:

```
(506, 14)
```

In [30]:

```
df.tail()
```

Out[30]:

|  | crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | b |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **501** | 0.06263 | 0.0 | 11.93 | 0 | 0.573 | 6.593 | 69.1 | 2.4786 | 1 | 273 | 21.0 | 391.99 |
| **502** | 0.04527 | 0.0 | 11.93 | 0 | 0.573 | 6.120 | 76.7 | 2.2875 | 1 | 273 | 21.0 | 396.90 |
| **503** | 0.06076 | 0.0 | 11.93 | 0 | 0.573 | 6.976 | 91.0 | 2.1675 | 1 | 273 | 21.0 | 396.90 |
| **504** | 0.10959 | 0.0 | 11.93 | 0 | 0.573 | 6.794 | 89.3 | 2.3889 | 1 | 273 | 21.0 | 393.45 |
| **505** | 0.04741 | 0.0 | 11.93 | 0 | 0.573 | 6.030 | 80.8 | 2.5050 | 1 | 273 | 21.0 | 396.90 |

In [31]:

```
df.corr()
```

Out[31]:

| | crim | zn | indus | chas | nox | rm | age |
|---|---|---|---|---|---|---|---|
| **crim** | 1.000000 | -0.200469 | 0.406583 | -0.055892 | 0.420972 | -0.219433 | 0.352734 |
| **zn** | -0.200469 | 1.000000 | -0.533828 | -0.042697 | -0.516604 | 0.311173 | -0.569537 |
| **indus** | 0.406583 | -0.533828 | 1.000000 | 0.062938 | 0.763651 | -0.394193 | 0.644779 |
| **chas** | -0.055892 | -0.042697 | 0.062938 | 1.000000 | 0.091203 | 0.091468 | 0.086518 |
| **nox** | 0.420972 | -0.516604 | 0.763651 | 0.091203 | 1.000000 | -0.302751 | 0.731470 |
| **rm** | -0.219433 | 0.311173 | -0.394193 | 0.091468 | -0.302751 | 1.000000 | -0.240286 |
| **age** | 0.352734 | -0.569537 | 0.644779 | 0.086518 | 0.731470 | -0.240286 | 1.000000 |
| **dis** | -0.379670 | 0.664408 | -0.708027 | -0.099176 | -0.769230 | 0.203507 | -0.747881 |
| **rad** | 0.625505 | -0.311948 | 0.595129 | -0.007368 | 0.611441 | -0.210718 | 0.456022 |
| **tax** | 0.582764 | -0.314563 | 0.720760 | -0.035587 | 0.668023 | -0.292794 | 0.506456 |
| **ptratio** | 0.289946 | -0.391679 | 0.383248 | -0.121515 | 0.188933 | -0.357612 | 0.261515 |
| **b** | -0.385064 | 0.175520 | -0.356977 | 0.048788 | -0.380051 | 0.128107 | -0.273534 |
| **lstat** | 0.455621 | -0.412995 | 0.603800 | -0.053929 | 0.590879 | -0.615721 | 0.602339 |
| **medv** | -0.388305 | 0.360445 | -0.483725 | 0.175260 | -0.427321 | 0.696169 | -0.376955 |

In [32]:

```
df.isnull()
```

Out[32]:

| | crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | b | l |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | False | False | False | False | False | False | False | False | False | False | False | False | F |
| **1** | False | False | False | False | False | False | False | False | False | False | False | False | F |
| **2** | False | False | False | False | False | False | False | False | False | False | False | False | F |
| **3** | False | False | False | False | False | False | False | False | False | False | False | False | F |
| **4** | False | False | False | False | False | False | False | False | False | False | False | False | F |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | |
| **501** | False | False | False | False | False | False | False | False | False | False | False | False | F |
| **502** | False | False | False | False | False | False | False | False | False | False | False | False | F |
| **503** | False | False | False | False | False | False | False | False | False | False | False | False | F |
| **504** | False | False | False | False | False | False | False | False | False | False | False | False | F |
| **505** | False | False | False | False | False | False | False | False | False | False | False | False | F |

506 rows × 14 columns

In [33]:

```
df.isnull().sum()
```

Out[33]:

```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         5
age        0
dis        0
rad        0
tax        0
ptratio    0
b          0
lstat      0
medv       0
dtype: int64
```

In [34]:

```
df['rm']=df['rm'].fillna(df['rm'].mean())
```

In [35]:

```
df.isnull().sum()
```

Out[35]:

```
crim       0
zn         0
indus      0
chas       0
nox        0
rm         0
age        0
dis        0
rad        0
tax        0
ptratio    0
b          0
lstat      0
medv       0
dtype: int64
```

In [36]:

```
df.head()
```

Out[36]:

| | crim | zn | indus | chas | nox | rm | age | dis | rad | tax | ptratio | b |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 0.00632 | 18.0 | 2.31 | 0 | 0.538 | 6.575 | 65.2 | 4.0900 | 1 | 296 | 15.3 | 396.90 |
| **1** | 0.02731 | 0.0 | 7.07 | 0 | 0.469 | 6.421 | 78.9 | 4.9671 | 2 | 242 | 17.8 | 396.90 |
| **2** | 0.02729 | 0.0 | 7.07 | 0 | 0.469 | 7.185 | 61.1 | 4.9671 | 2 | 242 | 17.8 | 392.83 |
| **3** | 0.03237 | 0.0 | 2.18 | 0 | 0.458 | 6.998 | 45.8 | 6.0622 | 3 | 222 | 18.7 | 394.63 |
| **4** | 0.06905 | 0.0 | 2.18 | 0 | 0.458 | 7.147 | 54.2 | 6.0622 | 3 | 222 | 18.7 | 396.90 |

In [37]:

```
X=df.drop('medv',axis=1)
y=df['medv']
df.dropna(inplace= True)
print(X)
print(y)
```

```
        crim    zn  indus  chas    nox     rm   age     dis  rad  tax  \
0    0.00632  18.0   2.31     0  0.538  6.575  65.2  4.0900    1  296
1    0.02731   0.0   7.07     0  0.469  6.421  78.9  4.9671    2  242
2    0.02729   0.0   7.07     0  0.469  7.185  61.1  4.9671    2  242
3    0.03237   0.0   2.18     0  0.458  6.998  45.8  6.0622    3  222
4    0.06905   0.0   2.18     0  0.458  7.147  54.2  6.0622    3  222
..       ...   ...    ...   ...    ...    ...   ...     ...  ...  ...
501  0.06263   0.0  11.93     0  0.573  6.593  69.1  2.4786    1  273
502  0.04527   0.0  11.93     0  0.573  6.120  76.7  2.2875    1  273
503  0.06076   0.0  11.93     0  0.573  6.976  91.0  2.1675    1  273
504  0.10959   0.0  11.93     0  0.573  6.794  89.3  2.3889    1  273
505  0.04741   0.0  11.93     0  0.573  6.030  80.8  2.5050    1  273

     ptratio       b  lstat
0       15.3  396.90   4.98
1       17.8  396.90   9.14
2       17.8  392.83   4.03
3       18.7  394.63   2.94
4       18.7  396.90   5.33
..       ...     ...    ...
501     21.0  391.99   9.67
502     21.0  396.90   9.08
503     21.0  396.90   5.64
504     21.0  393.45   6.48
505     21.0  396.90   7.88

[506 rows x 13 columns]
0      24.0
1      21.6
2      34.7
3      33.4
4      36.2
       ...
501    22.4
502    20.6
503    23.9
504    22.0
505    11.9
Name: medv, Length: 506, dtype: float64
```

In [38]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_stat
```

In [39]:

```
model=LinearRegression()
model.fit(X_train,y_train)
```

Out[39]:

```
▾ LinearRegression
LinearRegression()
```

In [40]:

```
print("Shape of X_train: ",X_train.shape)
print("Shape of X_test: ", X_test.shape)
print("Shape of y_train: ",y_train.shape)
print("Shape of y_test",y_test.shape)
```

```
Shape of X_train:  (404, 13)
Shape of X_test:  (102, 13)
Shape of y_train:  (404,)
Shape of y_test (102,)
```

In [41]:

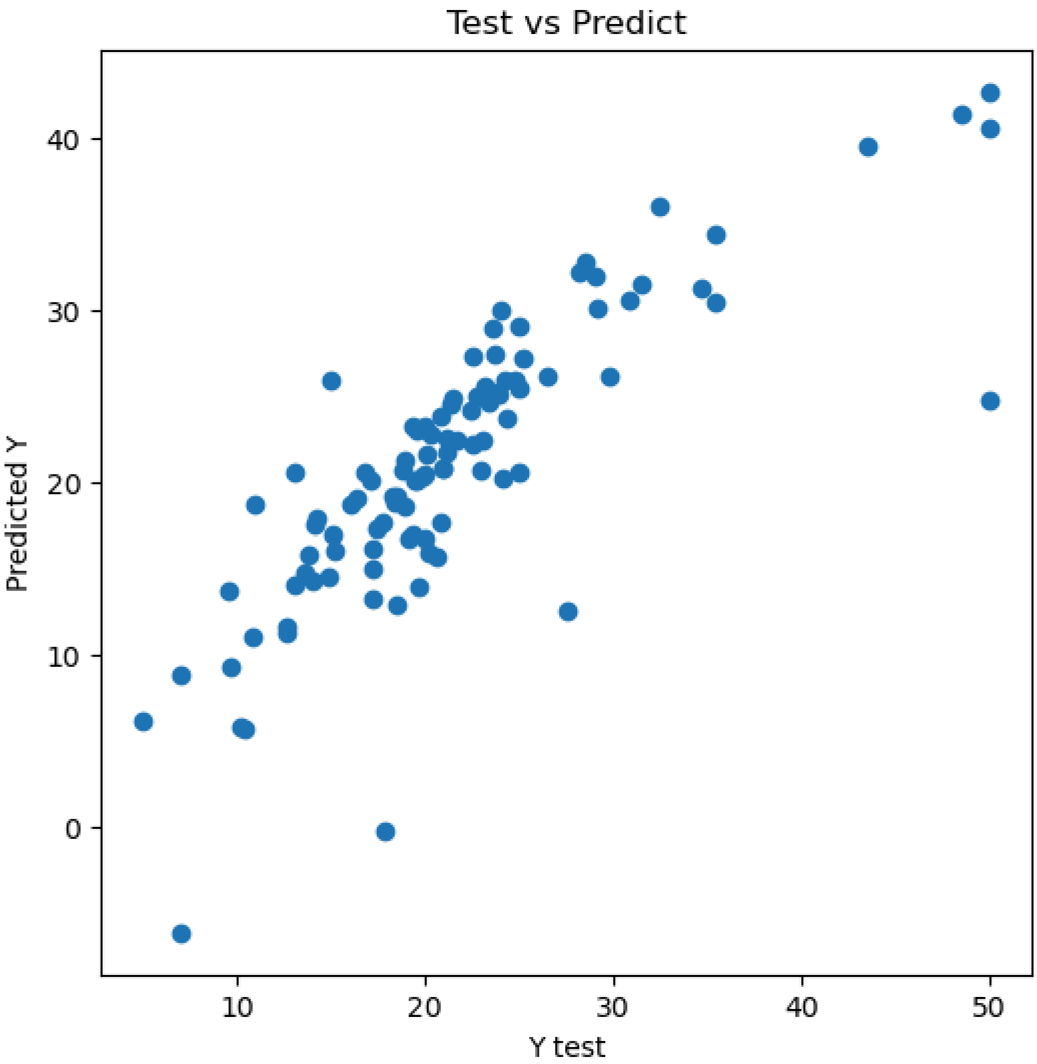
```
pred = model.predict(X_test)
```

In [42]:

```
plt.figure(figsize=(6,6));
plt.scatter(y_test,pred);
plt.xlabel('Y test')
plt.ylabel('Predicted Y')
plt.title('Test vs Predict')
```

Out[42]:

```
Text(0.5, 1.0, 'Test vs Predict')
```



In [43]:

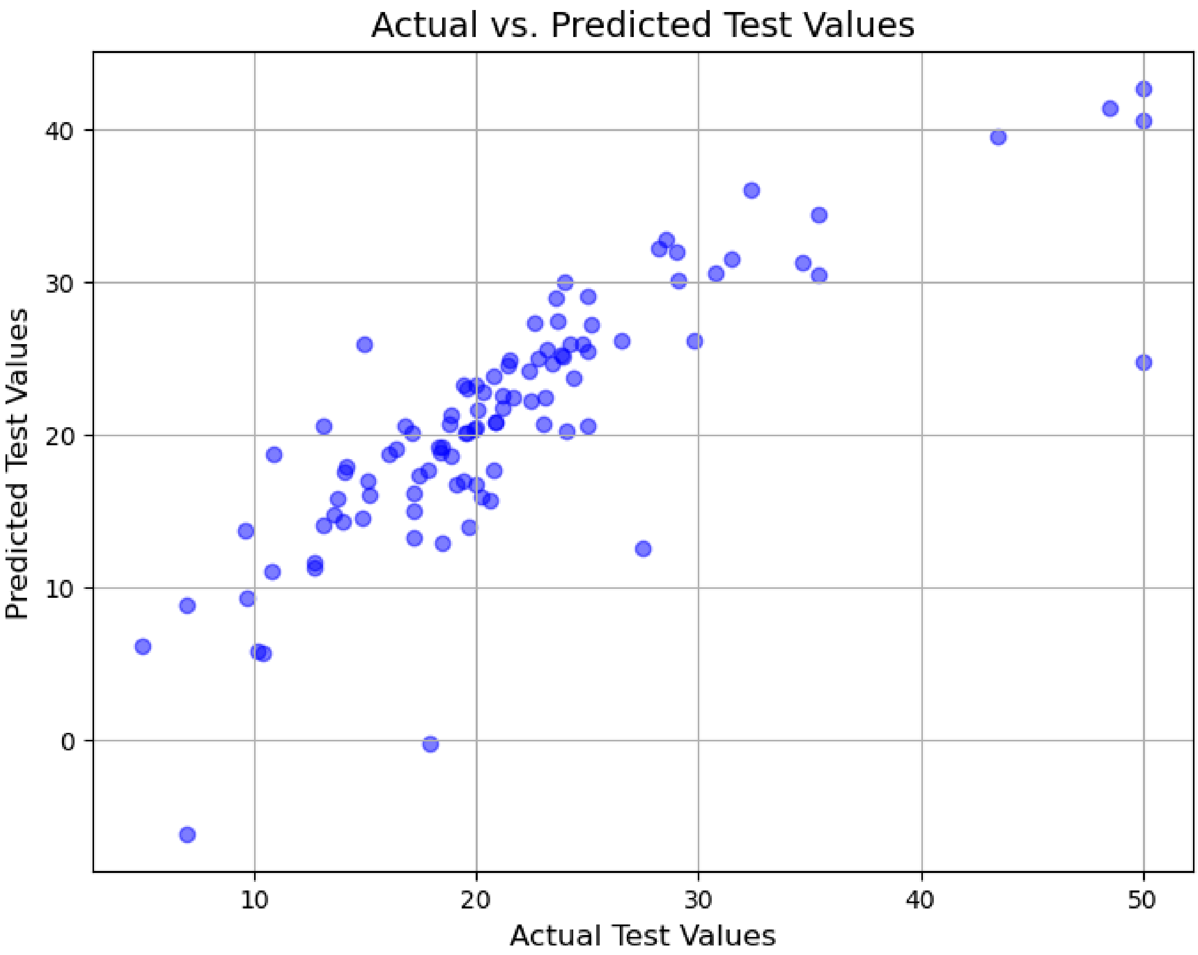
```
from sklearn import metrics

print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test,
Mean Absolute Error: 3.2064039639004025
Mean Squared Error: 24.404825188146653
Root Mean Squared Error: 4.940124005341025
```

In [44]:

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))  # Adjust figure size for better visibility
plt.scatter(y_test, pred, alpha=0.5, color='blue')  # Add transparency and s
plt.xlabel('Actual Test Values', fontsize=12)  # Customize x-axis label
plt.ylabel('Predicted Test Values', fontsize=12)  # Customize y-axis label
plt.title('Actual vs. Predicted Test Values', fontsize=14)  # Customize plot
plt.grid(True)  # Add grid lines
plt.show()
```



In [46]:

```python
r2 = r2_score(y_test, pred)
print('R-squared Score:', r2)
```

```
R-squared Score: 0.6672089705941856
```

In [ ]: