In [10]:
```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

df = pd.read_csv("BostonHousing.csv")
df.head(5)
df.isnull().sum()
df['rm'].fillna(df['rm'].mean(), inplace=True) # do this if missing values are there

X=df[['rm','lstat','ptratio']]
y=df['medv']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

y_pred = model.predict(X_test)

print("Predicted house prices:\n", y_pred)
print("Mean Squared Error:", mean_squared_error(y_test, y_pred))
print("R-squared Score:", r2_score(y_test, y_pred))

plt.figure(figsize=(8, 6))

plt.scatter(y_test, y_pred, color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', linestyle='--')  # Diagonal reference line

plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.title("Actual vs Predicted Home Prices")
plt.grid(True)
plt.show()
```

In [10]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error,r2_score
```

In [11]:
```
df=pd.read_csv('BostonHousing.csv')
```

In [12]:
```
df.head(5)
```

Out[12]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat	medv
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

In [31]:
```
df.isnull().sum()
```

Out[31]:
```
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 [32]:
```
df['rm'].fillna(df['rm'].mean(), inplace=True)
```

```
C:\Users\Chatura Karankal\AppData\Local\Temp\ipykernel_23520\3418382971.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original object.


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

In [33]:
```
df.isnull().sum()
```

Out[33]:
```
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 [34]:
```
X=df[['rm','lstat','ptratio']]
y=df['medv']
```

In [35]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=42)
```

In [36]:
```
model=LinearRegression()
model.fit(X_train,y_train)
```

Out[36]:
```
LinearRegression()
```

In [37]:
```
y_pred=model.predict(X_test)
```

In [38]:
```
mean_squared_error(y_test,y_pred)
```

Out[38]:
```
27.1454450741516
```

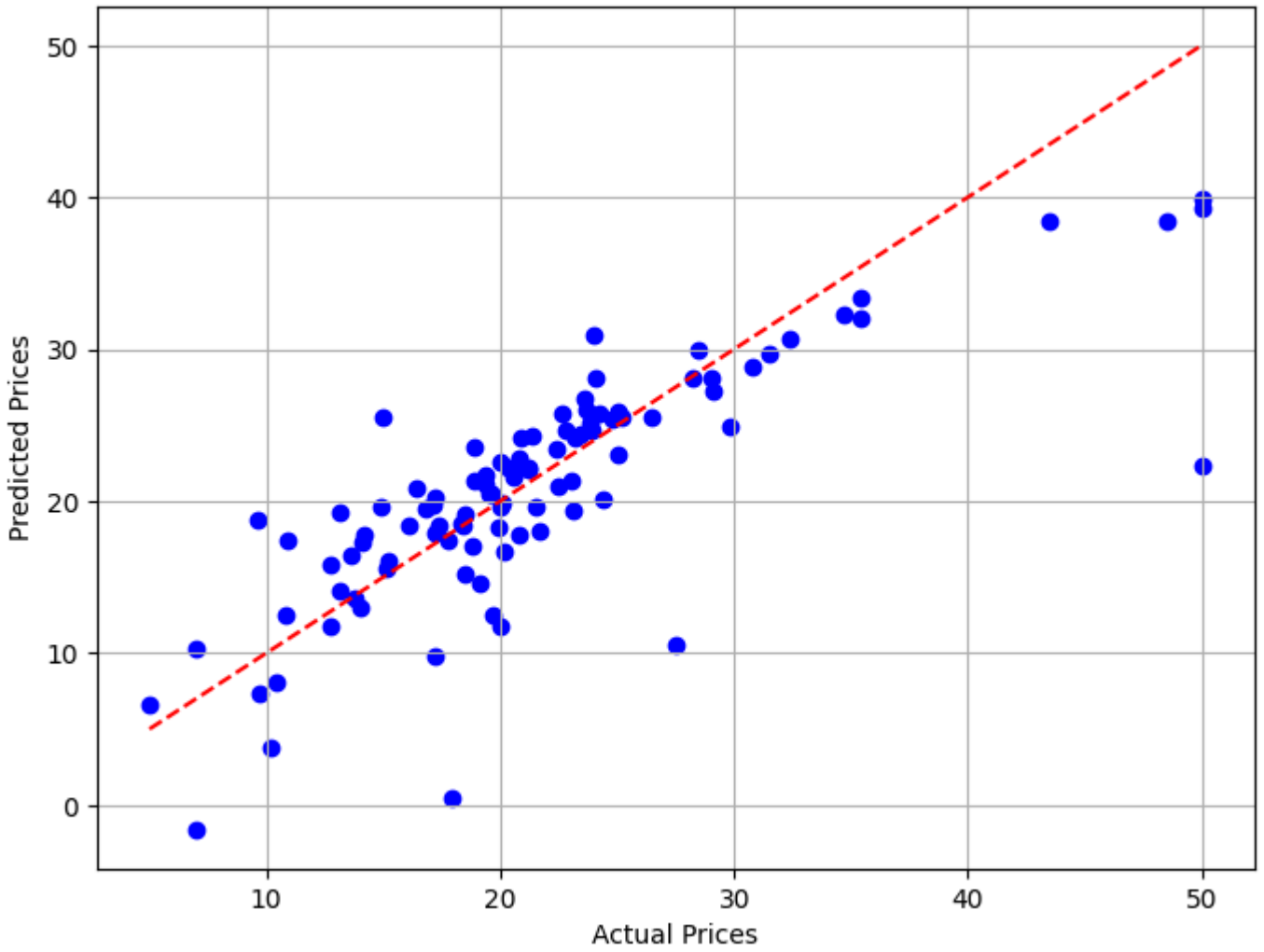
In [39]:
```
r2_score(y_test,y_pred)
```

Out[39]:
```
0.6298371104787273
```

In [45]:
```
plt.figure(figsize=(8, 6))

plt.scatter(y_test,y_pred,color='blue')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], color='red', linestyle='--')  # Diagonal reference line

plt.xlabel("Actual Prices")
plt.ylabel("Predicted Prices")
plt.grid(True)
plt.show()
```



In [ ]: