In [2]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import StandardScaler
```

In [3]:

```
df=pd.read_csv(r"C:\Users\pucha\Downloads\used_cars_data.csv")
df
```

Out[3]:

| | S.No. | Name | Location | Year | Kilometers_Driven | Fuel_Type | Transmission | Owner_Type | Mileage |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 0 | Maruti Wagon R LXI CNG | Mumbai | 2010 | 72000 | CNG | Manual | First | 26.6 km/kg |
| **1** | 1 | Hyundai Creta 1.6 CRDi SX Option | Pune | 2015 | 41000 | Diesel | Manual | First | 19.67 kmpl |
| **2** | 2 | Honda Jazz V | Chennai | 2011 | 46000 | Petrol | Manual | First | 18.2 kmpl |
| **3** | 3 | Maruti Ertiga VDI | Chennai | 2012 | 87000 | Diesel | Manual | First | 20.77 kmpl |
| **4** | 4 | Audi A4 New 2.0 TDI Multitronic | Coimbatore | 2013 | 40670 | Diesel | Automatic | Second | 15.2 kmpl |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **7248** | 7248 | Volkswagen Vento Diesel Trendline | Hyderabad | 2011 | 89411 | Diesel | Manual | First | 20.54 kmpl |
| **7249** | 7249 | Volkswagen Polo GT TSI | Mumbai | 2015 | 59000 | Petrol | Automatic | First | 17.21 kmpl |
| **7250** | 7250 | Nissan Micra Diesel XV | Kolkata | 2012 | 28000 | Diesel | Manual | First | 23.08 kmpl |
| **7251** | 7251 | Volkswagen Polo GT TSI | Pune | 2013 | 52262 | Petrol | Automatic | Third | 17.2 kmpl |
| **7252** | 7252 | Mercedes-Benz E-Class 2009-2013 E 220 CDI Avan... | Kochi | 2014 | 72443 | Diesel | Automatic | First | 10.0 kmpl |

7253 rows × 14 columns

In [4]:

```
pd.set_option('display.max_rows',10000000000)
pd.set_option('display.max_columns',10000000000)
pd.set_option('display.width',95)
```

In [5]:

```
print('The DataFrame has %d Rows and %d Columns'%(df.shape))
```

```
The DataFrame has 7253 Rows and 14 Columns
```

In [6]:

```
df=df[['Year','Price']]
df.columns=['year','price']
df.head(10)
```

Out[6]:

| | year | price |
|---|---|---|
| **0** | 2010 | 1.75 |
| **1** | 2015 | 12.50 |
| **2** | 2011 | 4.50 |
| **3** | 2012 | 6.00 |
| **4** | 2013 | 17.74 |
| **5** | 2012 | 2.35 |
| **6** | 2013 | 3.50 |
| **7** | 2016 | 17.50 |
| **8** | 2013 | 5.20 |
| **9** | 2012 | 1.95 |

In [7]:

```
df.head()
```

Out[7]:

| | year | price |
|---|---|---|
| **0** | 2010 | 1.75 |
| **1** | 2015 | 12.50 |
| **2** | 2011 | 4.50 |
| **3** | 2012 | 6.00 |
| **4** | 2013 | 17.74 |

In [8]:

```
df.describe()
```

Out[8]:

| | year | price |
|---|---|---|
| **count** | 7253.000000 | 6019.000000 |
| **mean** | 2013.365366 | 9.479468 |
| **std** | 3.254421 | 11.187917 |
| **min** | 1996.000000 | 0.440000 |
| **25%** | 2011.000000 | 3.500000 |
| **50%** | 2014.000000 | 5.640000 |
| **75%** | 2016.000000 | 9.950000 |
| **max** | 2019.000000 | 160.000000 |

In [9]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7253 entries, 0 to 7252
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   year    7253 non-null   int64  
 1   price   6019 non-null   float64
dtypes: float64(1), int64(1)
memory usage: 113.5 KB
```

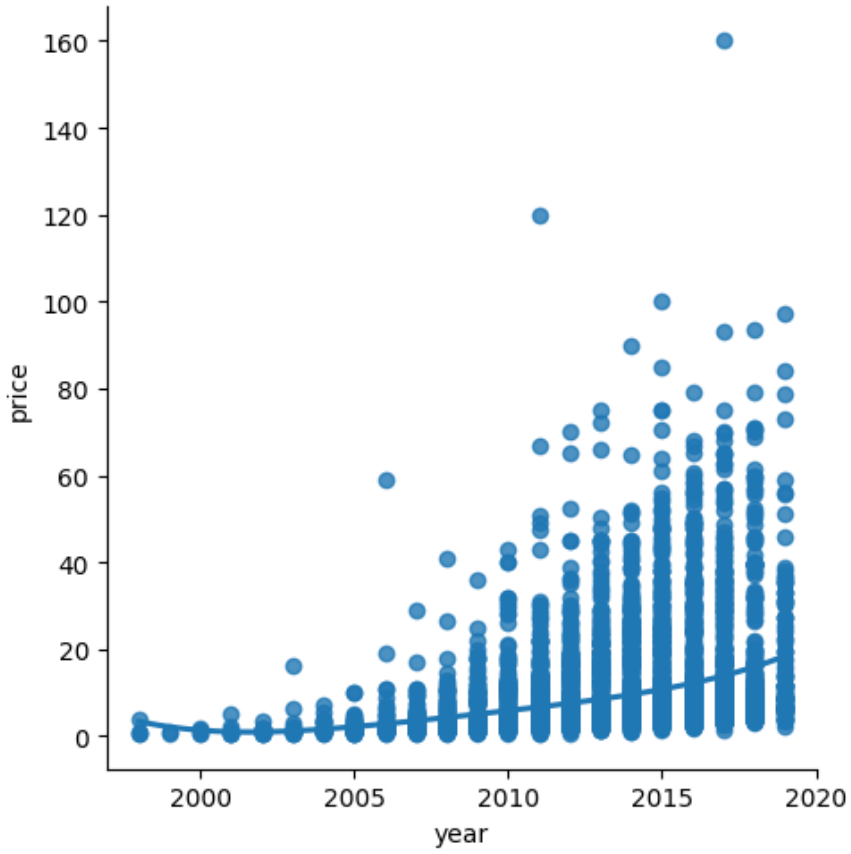
In [10]:

```
sns.lmplot(x="year",y="price",data=df,order=5,ci=None)
```

```
C:\Users\pucha\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\regr
ession.py:254: RankWarning: Polyfit may be poorly conditioned
  yhat = reg_func(x, y)
```

Out[10]:

```
<seaborn.axisgrid.FacetGrid at 0x2b4c4435c90>
```

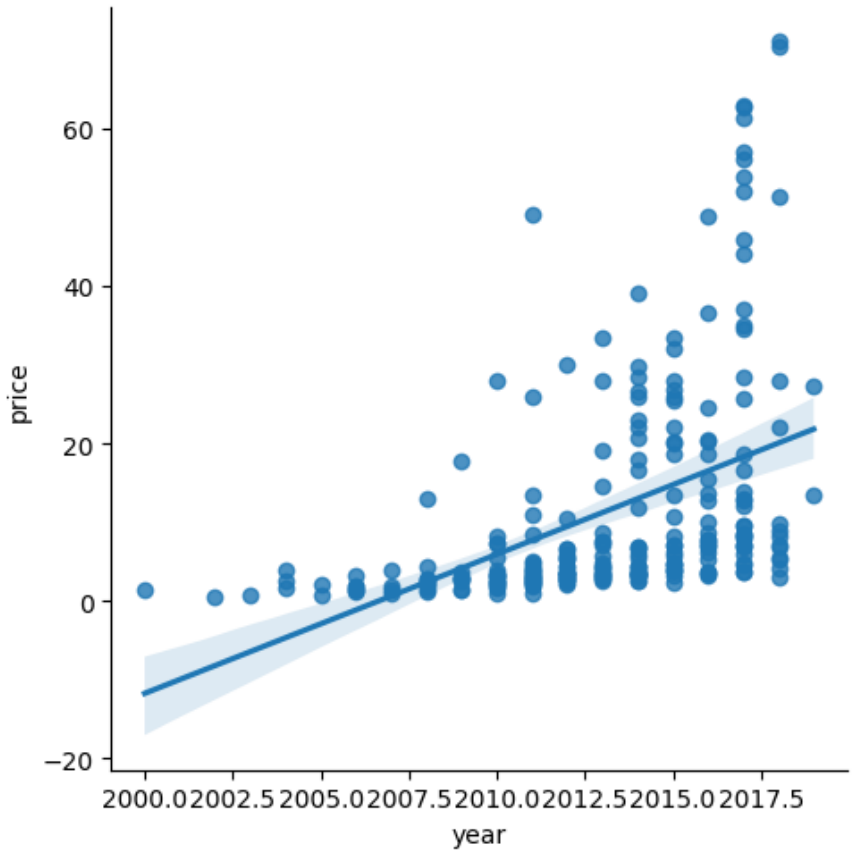


In [11]:

```
df500=df[:][250:500]
sns.lmplot(x="year",y="price",data=df500,order=1)
```

Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x2b4c4408a00>
```




In [12]:

```
x=np.array(df['year']).reshape(-1,1)
y=np.array(df['price']).reshape(-1,1)
df.dropna(inplace=True)
```

```
C:\Users\pucha\AppData\Local\Temp\ipykernel_464\1387402884.py:3: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.dropna(inplace=True)
```

In [14]:

```
features_matrix=df.iloc[:,0:34]
```

In [15]:

```
target_vector=df.iloc[:,-1]
```

In [16]:

```
print('The Features Matrix Has %d Rows and %d Columns'%(features_matrix.shape))
print('The Target Matrix Has %d Rows and %d Columns'%(np.array(target_vector).reshape(-1,1).shape))
```

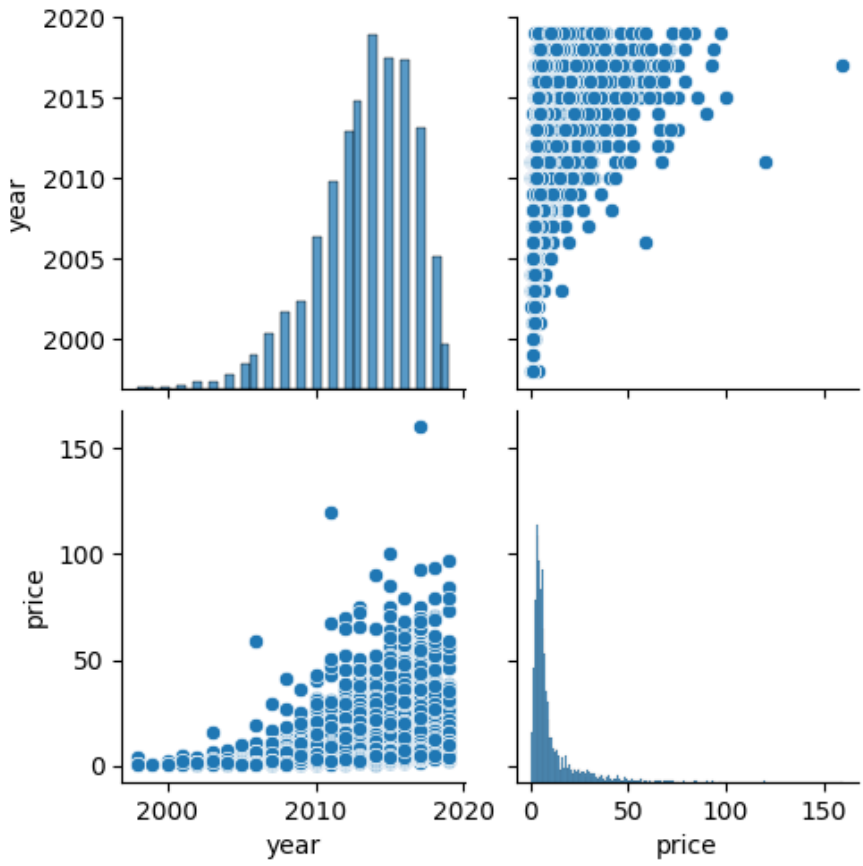
```
The Features Matrix Has 6019 Rows and 2 Columns
The Target Matrix Has 6019 Rows and 1 Columns
```

In [17]:

```
sns.pairplot(df)
```

Out[17]:

```
<seaborn.axisgrid.PairGrid at 0x2b4c4435870>
```

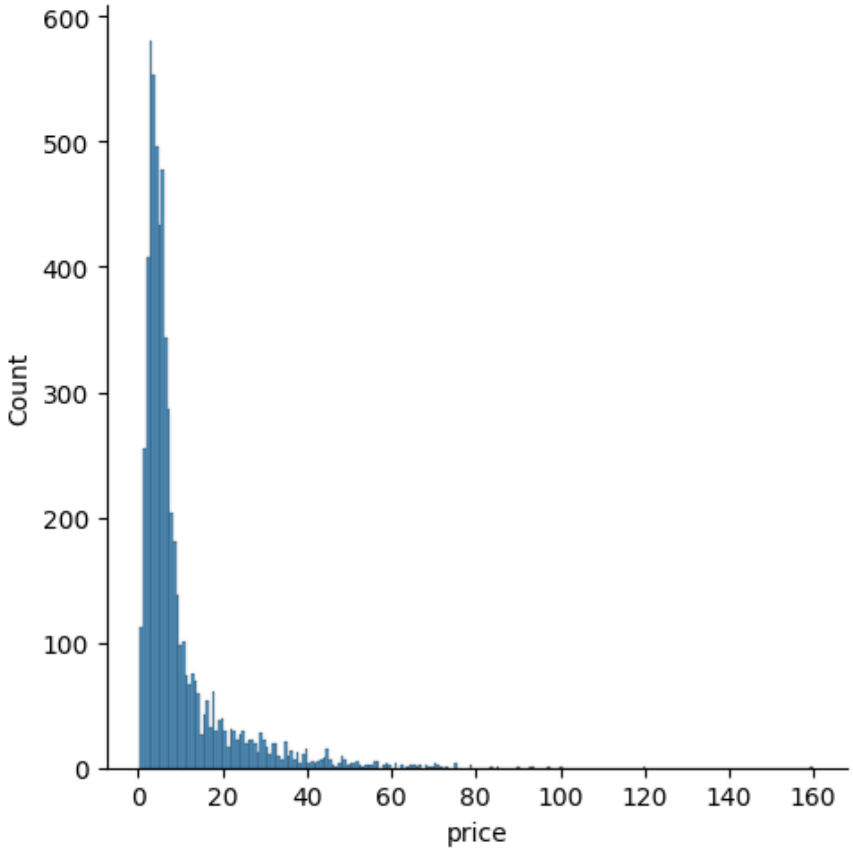


In [19]:

```
sns.displot(df['price'])
```

Out[19]:

```
<seaborn.axisgrid.FacetGrid at 0x2b4ca168c10>
```

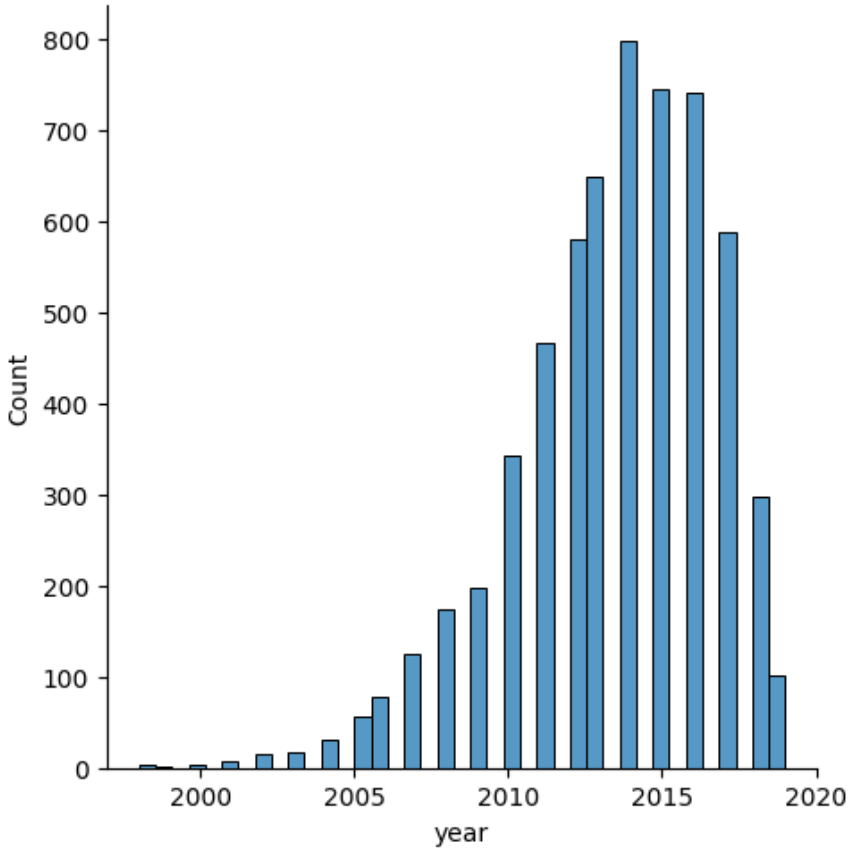


In [20]:

```
sns.displot(df['year'])
```

Out[20]:

```
<seaborn.axisgrid.FacetGrid at 0x2b4c425fac0>
```



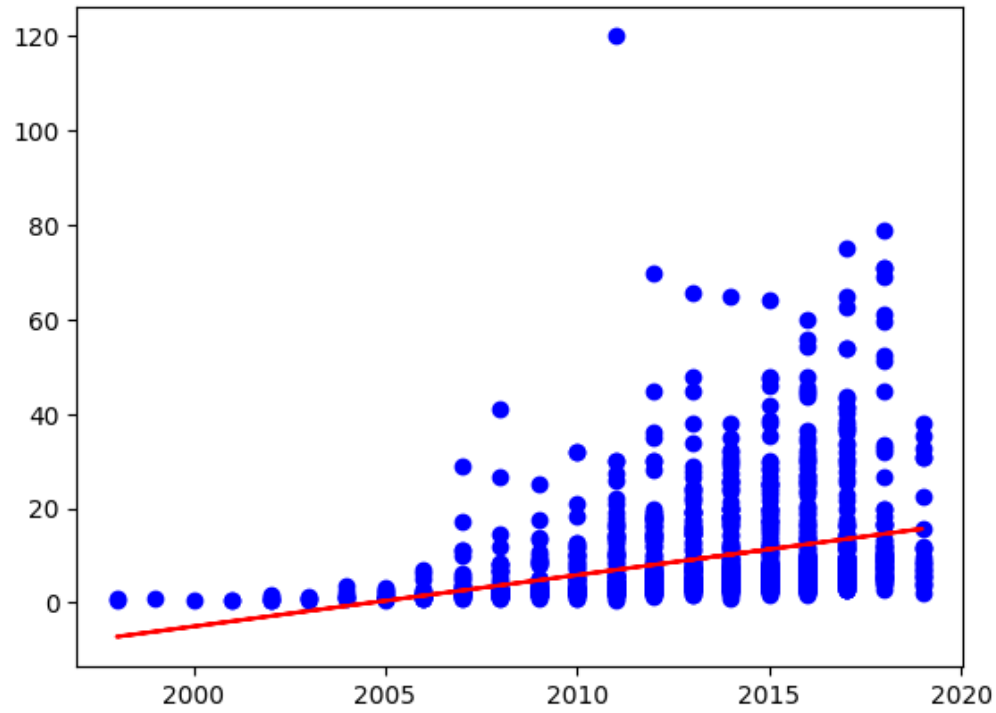
In [24]:

```
df500.fillna(method='ffill',inplace=True)
x=np.array(df['year']).reshape(-1,1)
y=np.array(df['price']).reshape(-1,1)
df.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression:",regr.score(x_test,y_test))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='r')
plt.show()
```

```
C:\Users\pucha\AppData\Local\Temp\ipykernel_464\4156448208.py:4: SettingWithCopyWarni
ng:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/us
er_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pand
as-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.dropna(inplace=True)

Regression: 0.07923369981620254
```




In [26]:

```
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
lm.fit(x_train,y_train)
```

Out[26]:

▾ LinearRegression

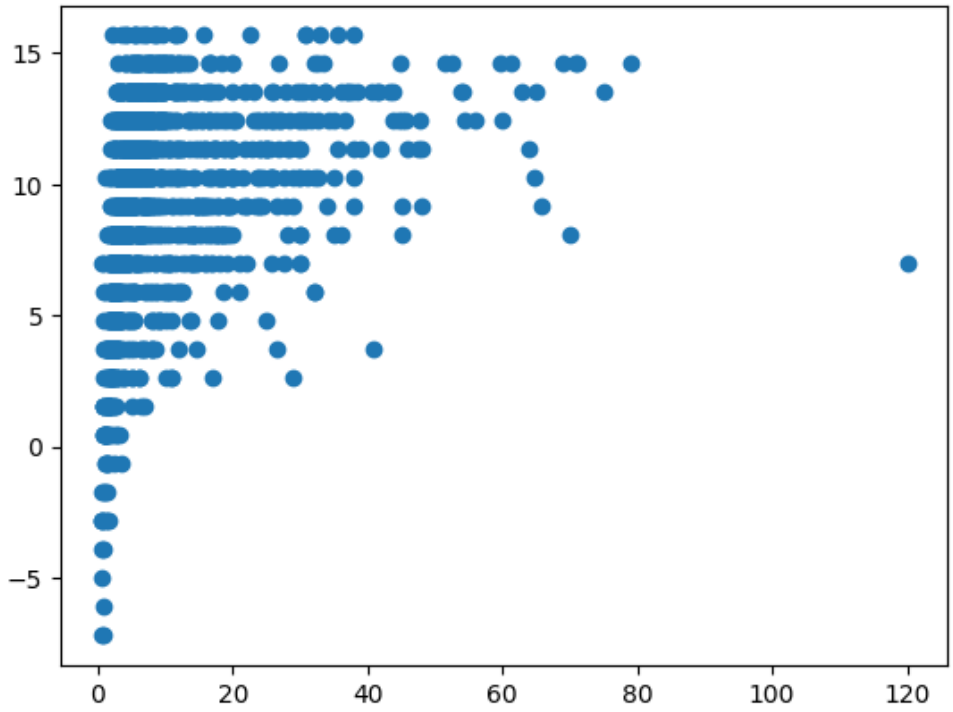
LinearRegression()

In [28]:

```
predictions=lm.predict(x_test)
plt.scatter(y_test,predictions)
```

Out[28]:

```
<matplotlib.collections.PathCollection at 0x2b4cc691d50>
```



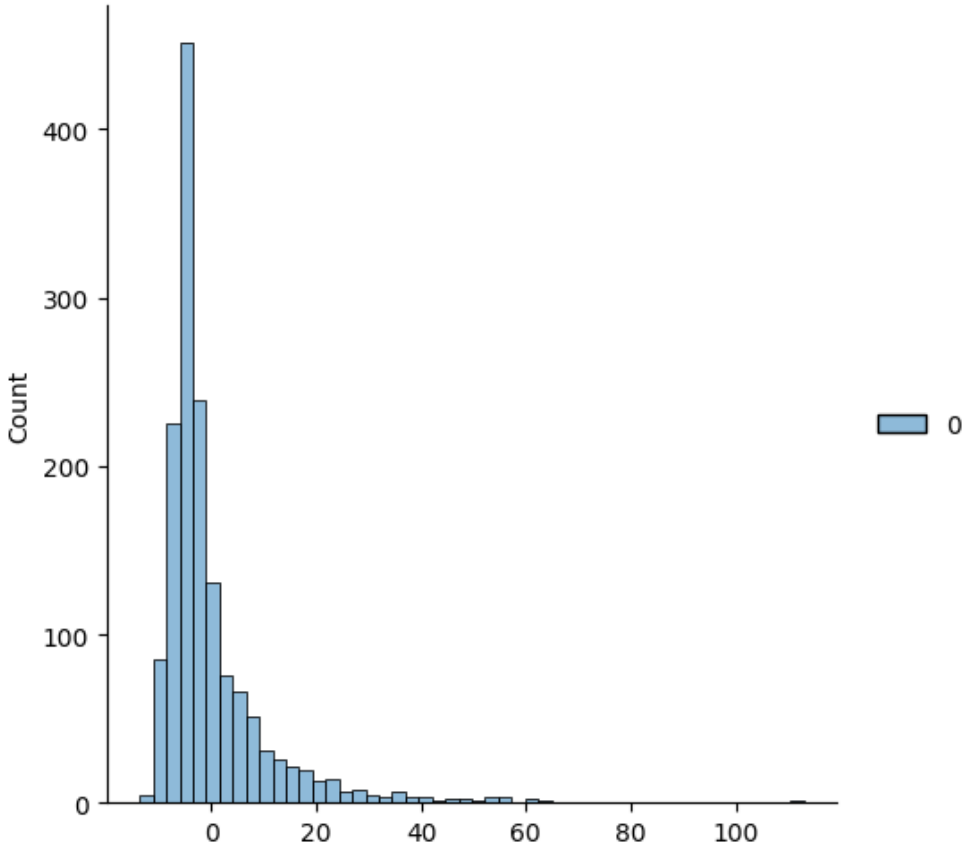

In [29]:

```
from sklearn import metrics
print('MAE:',metrics.mean_absolute_error(y_test,predictions))
print('MSE:',metrics.mean_squared_error(y_test,predictions))
print('RMSE:',np.sqrt(metrics.mean_squared_error(y_test,predictions)))
```

```
MAE: 6.7706565809809565
MSE: 107.96818426543761
RMSE: 10.390773997418941
```

In [30]:

```
sns.displot((y_test-predictions),bins=50);
```




In [31]:

```
df.columns
```

Out[31]:

```
Index(['year', 'price'], dtype='object')
```

In [32]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model=LinearRegression()
model.fit(x_train,y_train)
y_pred=model.predict(x_test)
r2=r2_score(y_test,y_pred)
print("R2_score:",r2)
```

```
R2_score: 0.07923369981620254
```

In [ ]:

```

```