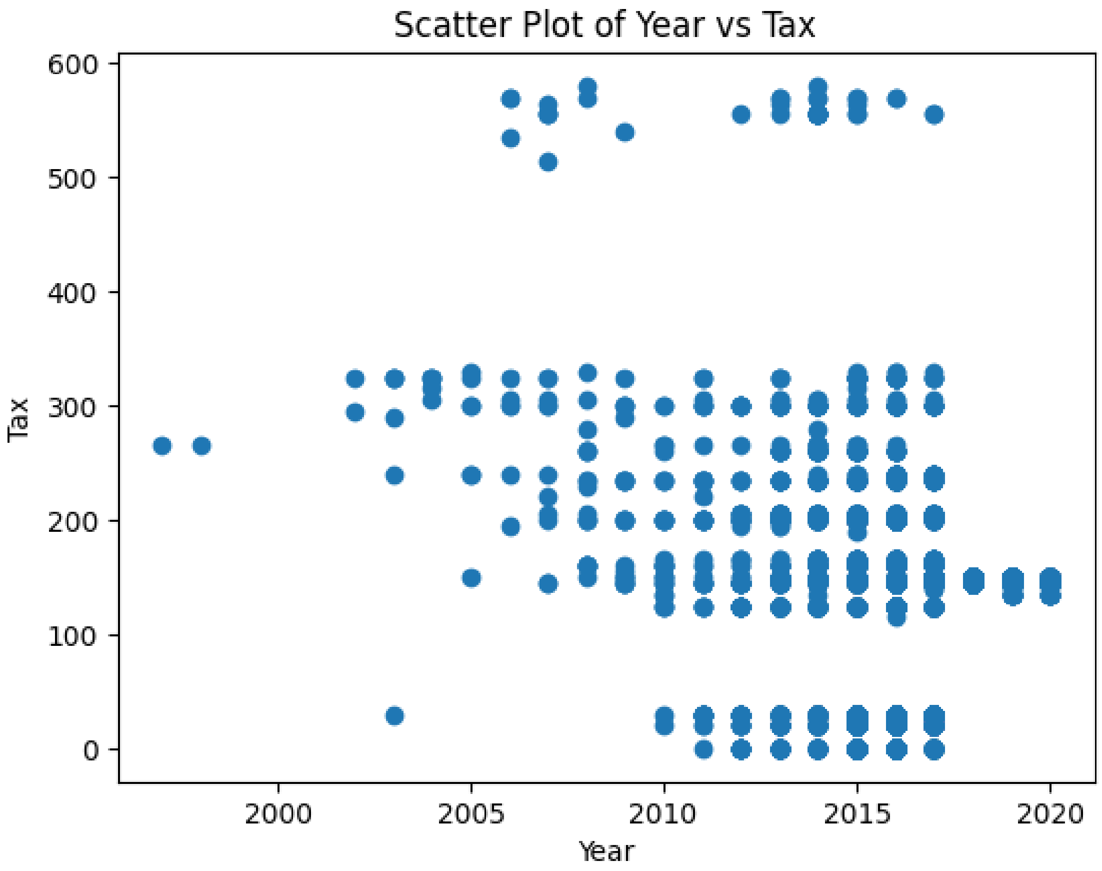
In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm

# Load and Reformat Data
data = pd.read_csv('C:/Users/user/Desktop/My learning/ClinSoft/audi.csv')

# Data Exploration
print(data.columns)
print(data.info())
print(data.describe())
```

```
Index(['model', 'year', 'price', 'transmission', 'mileage', 'fuelType',
'tax',
       'mpg', 'engineSize'],
      dtype='object')
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10668 entries, 0 to 10667
Data columns (total 9 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   model         10668 non-null  object 
 1   year          10668 non-null  int64  
 2   price         10668 non-null  int64  
 3   transmission  10668 non-null  object 
 4   mileage       10668 non-null  int64  
 5   fuelType      10668 non-null  object 
 6   tax           10668 non-null  int64  
 7   mpg           10668 non-null  float64
 8   engineSize    10668 non-null  float64
dtypes: float64(2), int64(4), object(3)
memory usage: 750.2+ KB
None
               year          price        mileage           tax  
mpg  \
count  10668.000000   10668.000000   10668.000000  10668.000000  10668.00
0000   
mean    2017.100675   22896.685039   24827.244001    126.011436     50.77
0022   
std        2.167494   11714.841888   23505.257205     67.170294     12.94
9782   
min     1997.000000    1490.000000       1.000000      0.000000     18.90
0000   
25%     2016.000000   15130.750000    5968.750000    125.000000     40.90
0000   
50%     2017.000000   20200.000000   19000.000000    145.000000     49.60
0000   
75%     2019.000000   27990.000000   36464.500000    145.000000     58.90
0000   
max     2020.000000  145000.000000  323000.000000    580.000000    188.30
0000   

         engineSize  
count  10668.000000  
mean       1.930709  
std        0.602957  
min        0.000000  
25%        1.500000  
50%        2.000000  
75%        2.000000  
max        6.300000  
```

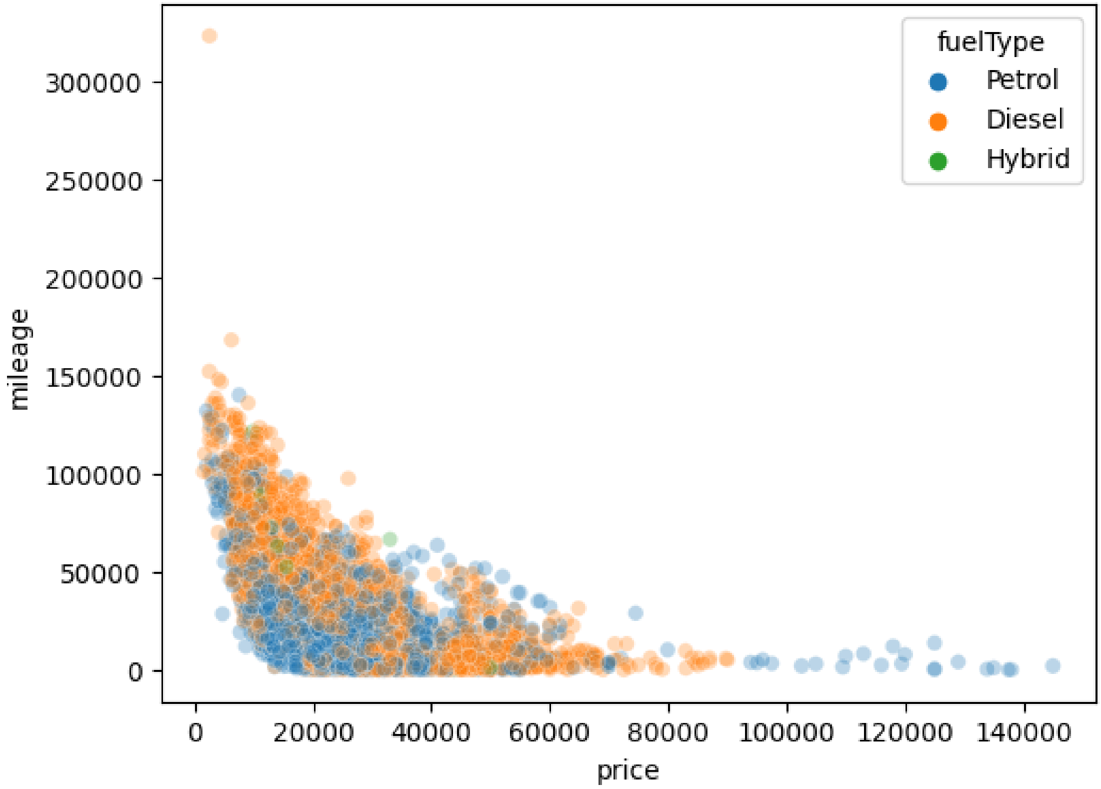
In [2]:

```
# Scatter Plot
plt.scatter(data['year'], data['tax'])
plt.xlabel('Year')
plt.ylabel('Tax')
plt.title('Scatter Plot of Year vs Tax')
plt.show()
```



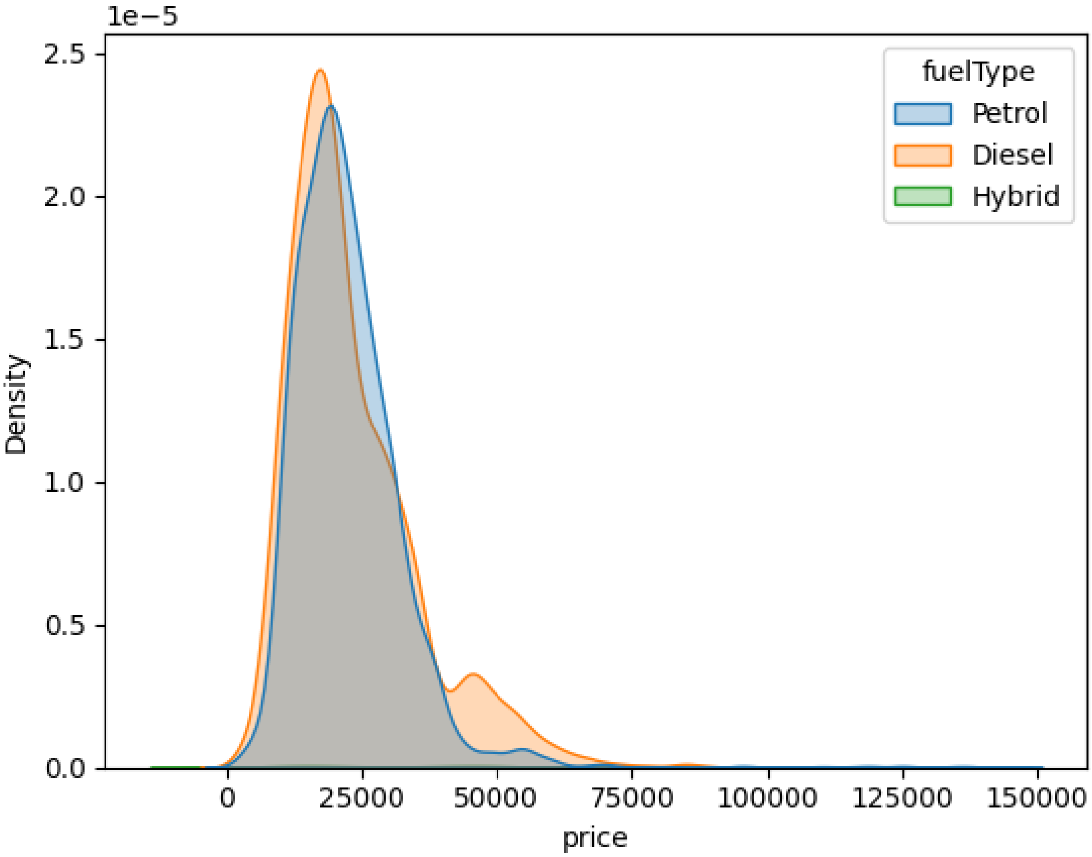
In [3]:

```
# Scatter Plot with Density
sns.scatterplot(x='price', y='mileage', hue='fuelType', data=data, alpha=0
plt.show()
```



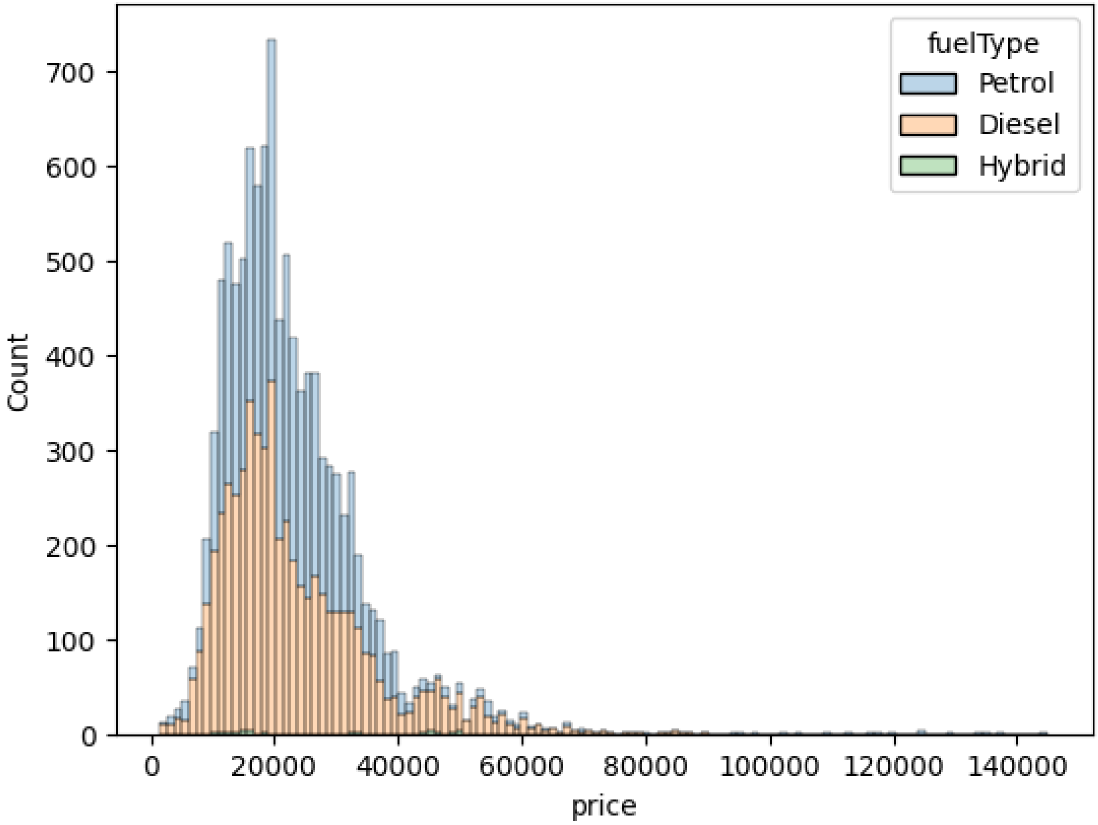
In [4]:

```
# Density Plot
sns.kdeplot(data=data, x='price', hue='fuelType', alpha=0.3, fill=True)
plt.show()
```



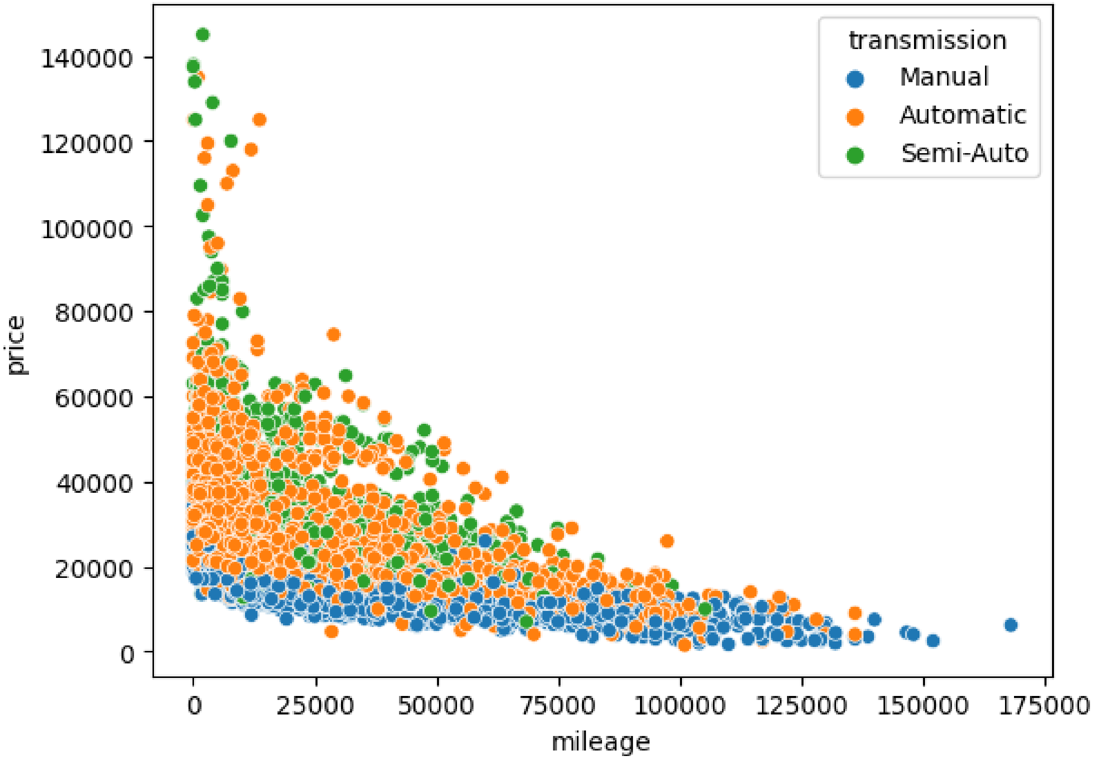
In [5]:

```
# Histogram
sns.histplot(data=data, x='price', hue='fuelType', alpha=0.3, multiple='st
plt.show()
```

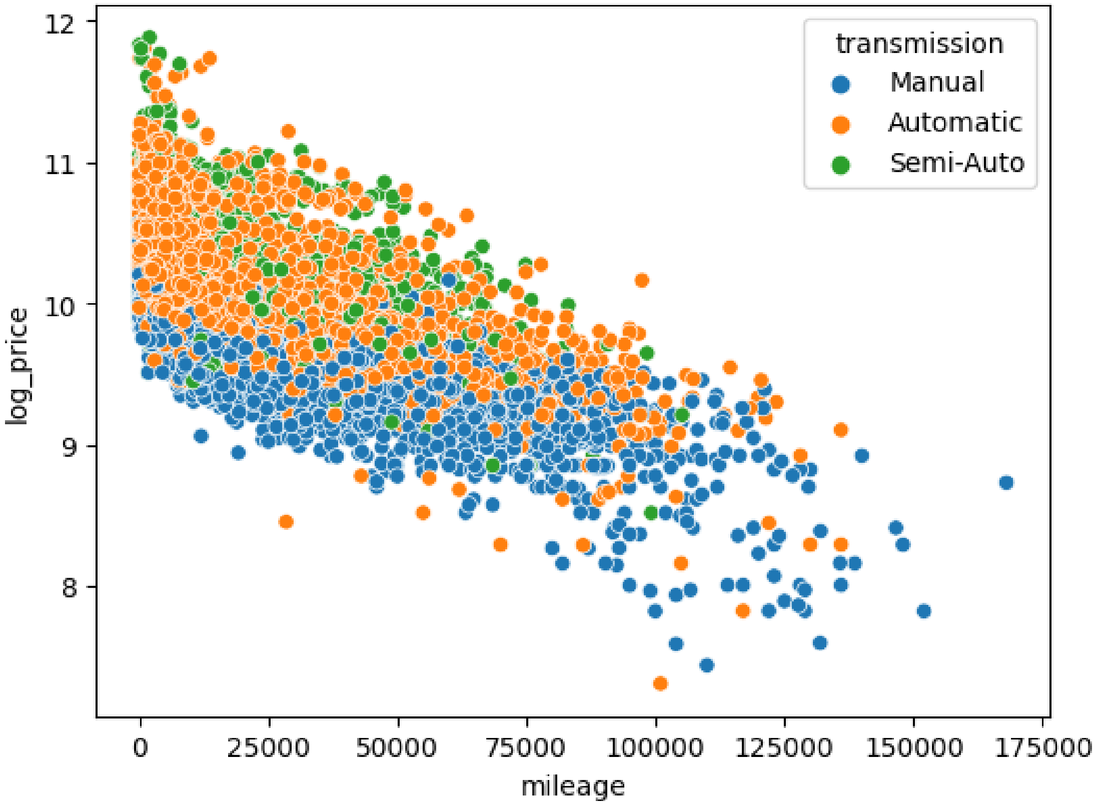


In [6]:

```
# Filtering and Plotting
filtered_data = data[data['mileage'] < 180000]
sns.scatterplot(data=filtered_data, x='mileage', y='price', hue='transmiss
plt.show()

# Log Transform and Plot
filtered_data['log_price'] = np.log(filtered_data['price'])
sns.scatterplot(data=filtered_data, x='mileage', y='log_price', hue='trans
plt.show()
```



```
C:\Users\user\AppData\Local\Temp\ipykernel_4512\2771945094.py:7: SettingW
ithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  filtered_data['log_price'] = np.log(filtered_data['price'])
```


transmission
Manual
Automatic
Semi-Auto
log_price
12
11
10
9
8
mileage
0
25000
50000
75000
100000
125000
150000
175000

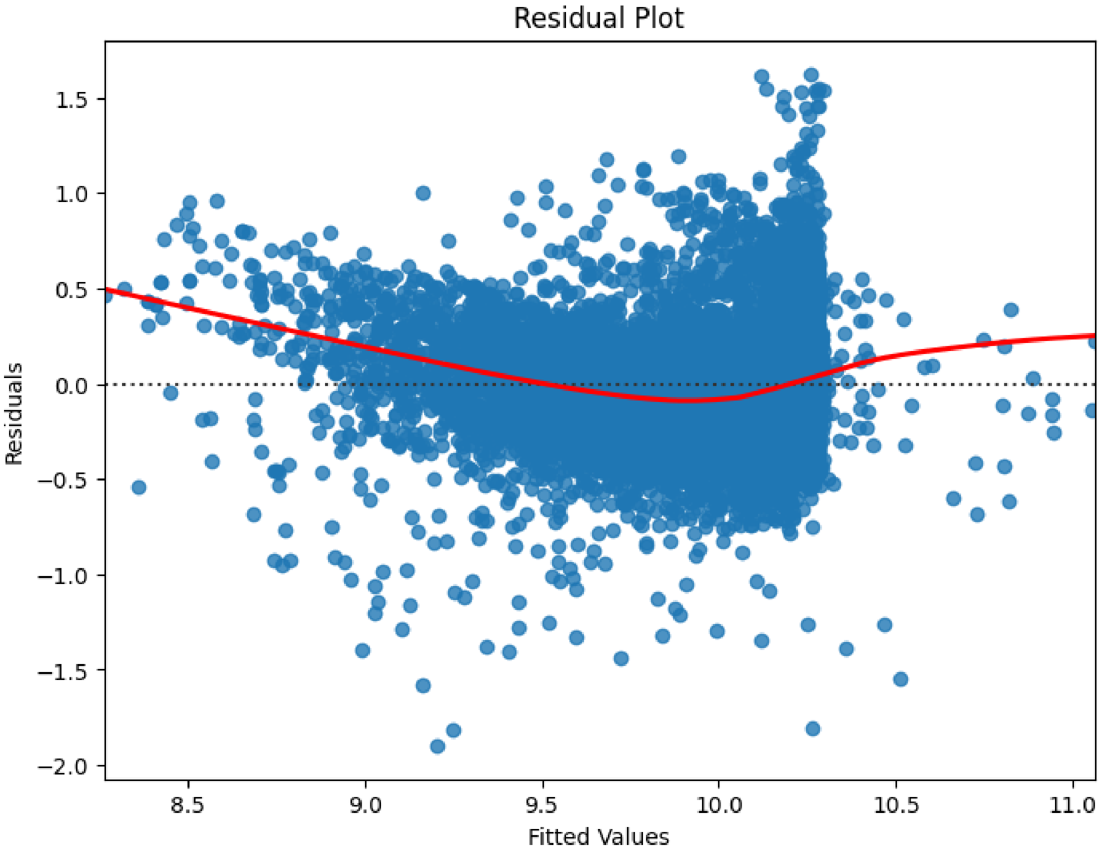
In [7]:

```
# Linear Regression
model = sm.OLS.from_formula('log_price ~ mileage + tax', data=filtered_dat
result = model.fit()
print(result.summary())
```

```
                            OLS Regression Results
=========================================================================
=====
Dep. Variable:              log_price   R-squared:
0.555
Model:                            OLS   Adj. R-squared:
0.555
Method:                 Least Squares   F-statistic:
6645.
Date:                Sat, 05 Aug 2023   Prob (F-statistic):
0.00
Time:                        22:53:06   Log-Likelihood:                -2
776.2
No. Observations:               10667   AIC:
5558.
Df Residuals:                   10664   BIC:
5580.
Df Model:                           2
Covariance Type:            nonrobust
=========================================================================
=====
                 coef    std err          t      P>|t|      [0.025
0.975]
-------------------------------------------------------------------------
-----
Intercept      9.9745      0.008   1288.946      0.000       9.959
9.990
mileage    -1.271e-05   1.32e-07    -96.116      0.000    -1.3e-05   -1.2
4e-05
tax            0.0021   4.59e-05     46.473      0.000       0.002
0.002
=========================================================================
=====
Omnibus:                      834.078   Durbin-Watson:
1.706
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             240
0.716
Skew:                           0.422   Prob(JB):
0.00
Kurtosis:                       5.165   Cond. No.                     8.6
7e+04
=========================================================================
=====

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is co
rrectly specified.
[2] The condition number is large, 8.67e+04. This might indicate that the
re are
strong multicollinearity or other numerical problems.
```

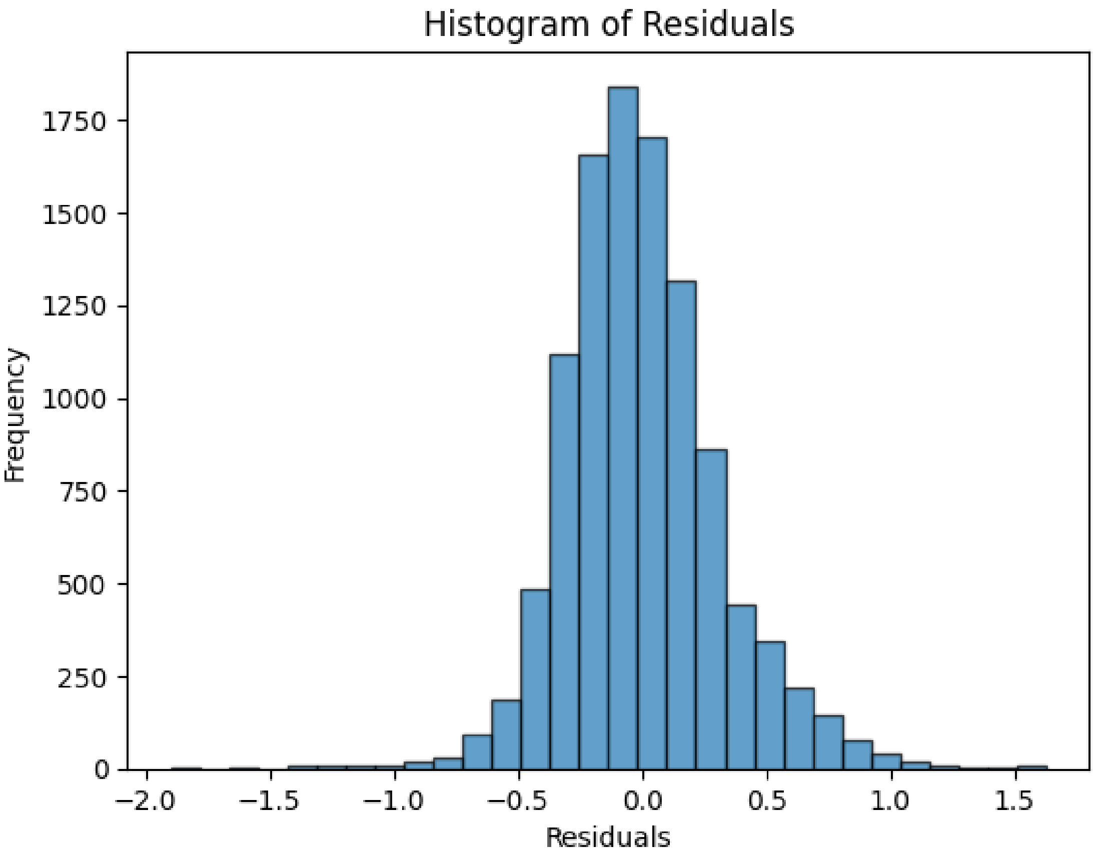
In [9]:

```
# Residual Analysis
fig, ax = plt.subplots(figsize=(8, 6))
sns.residplot(x=result.fittedvalues, y=result.resid, lowess=True, line_kws
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Residual Plot')
plt.show()
```



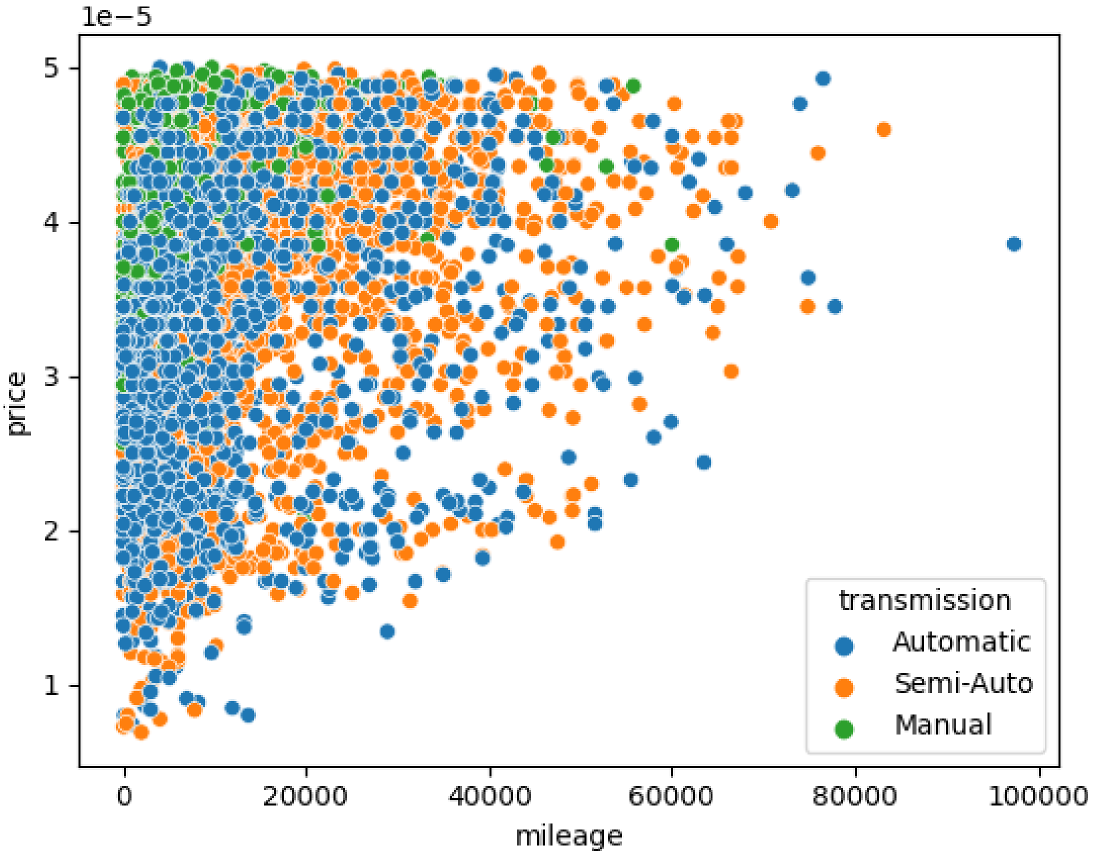
In [10]:

```
# Histogram of Residuals
plt.hist(result.resid, bins=30, edgecolor='k', alpha=0.7)
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.title('Histogram of Residuals')
plt.show()
```



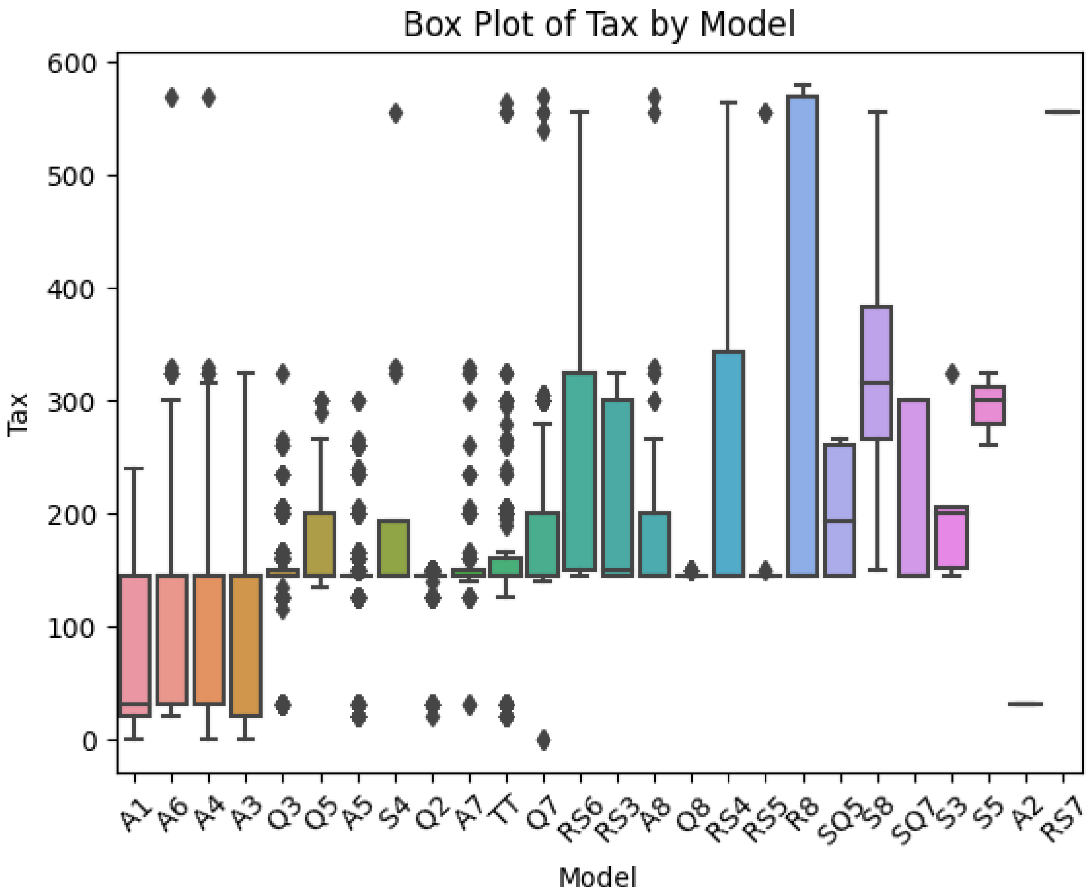
In [11]:

```
# Filtering Outliers
filtered_data = data[(data['mileage'] < 180000) & (data['price'] > 20000)]
sns.scatterplot(data=filtered_data, x='mileage', y=1/filtered_data['price'
plt.show()
```



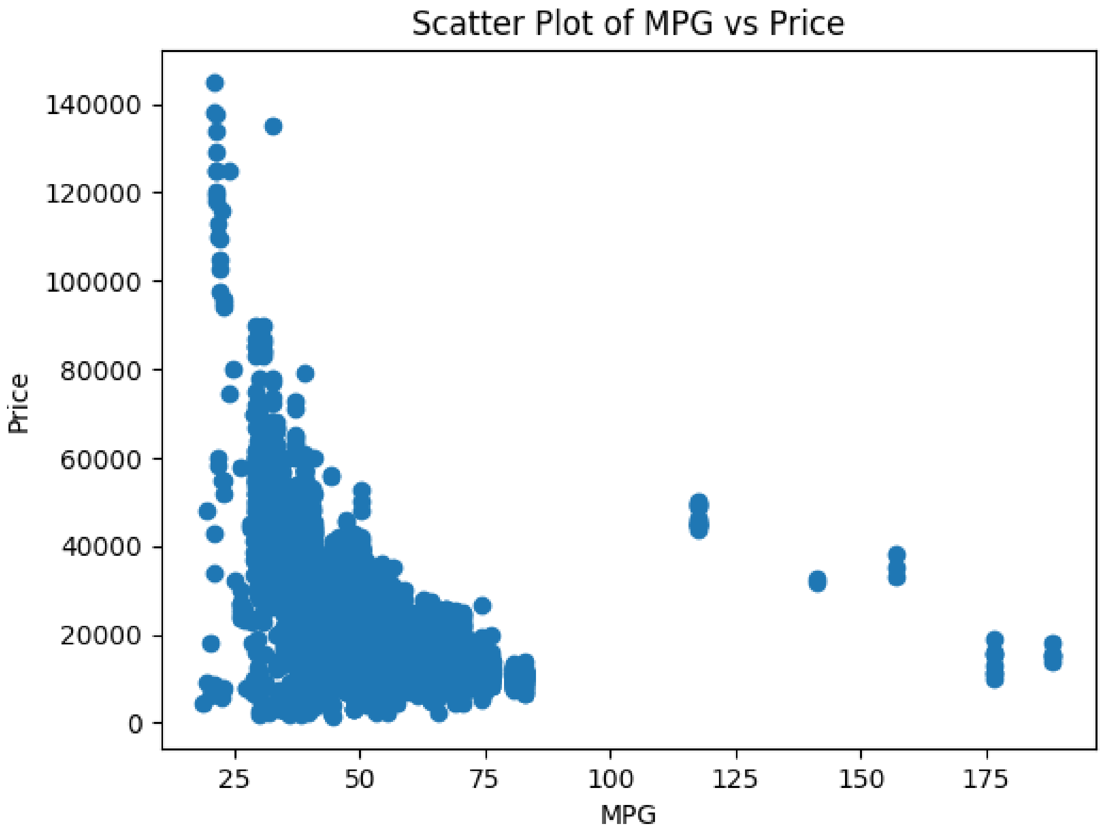
In [12]:

```
# Box Plot
sns.boxplot(data=data, x='model', y='tax')
plt.xticks(rotation=45)
plt.xlabel('Model')
plt.ylabel('Tax')
plt.title('Box Plot of Tax by Model')
plt.show()
```



In [13]:

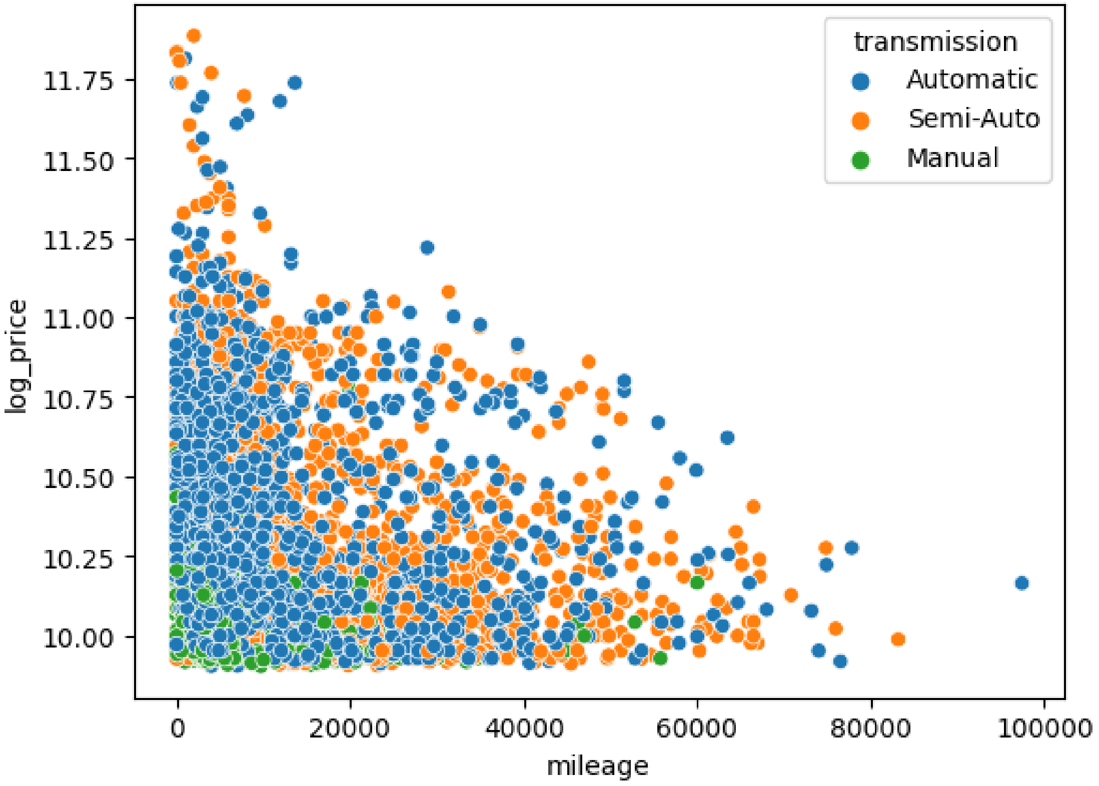
```
# Day 2
plt.scatter(data['mpg'], data['price'])
plt.xlabel('MPG')
plt.ylabel('Price')
plt.title('Scatter Plot of MPG vs Price')
plt.show()
```



In [19]:

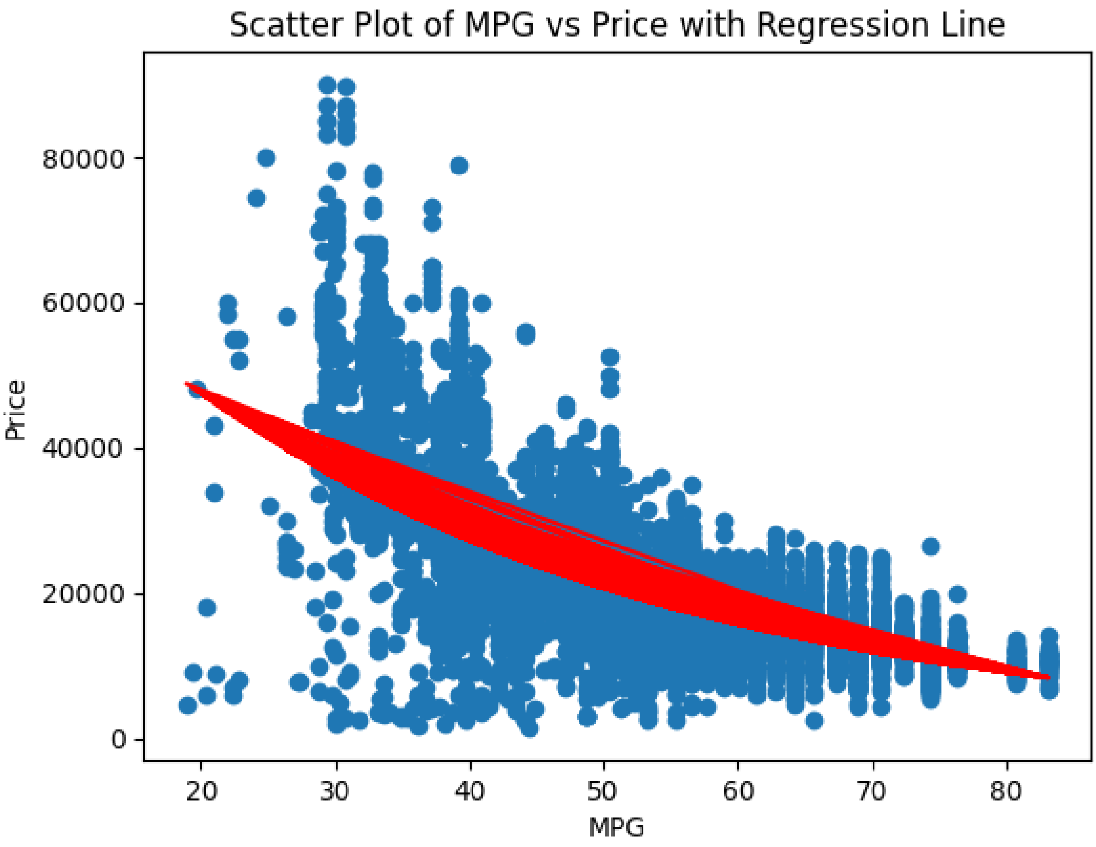
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import statsmodels.api as sm



filtered_data['log_price'] = np.log(filtered_data['price'])
sns.scatterplot(data=filtered_data, x='mileage', y='log_price', hue='trans
plt.show()
```

```
C:\Users\user\AppData\Local\Temp\ipykernel_4512\969365871.py:9: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  filtered_data['log_price'] = np.log(filtered_data['price'])
```



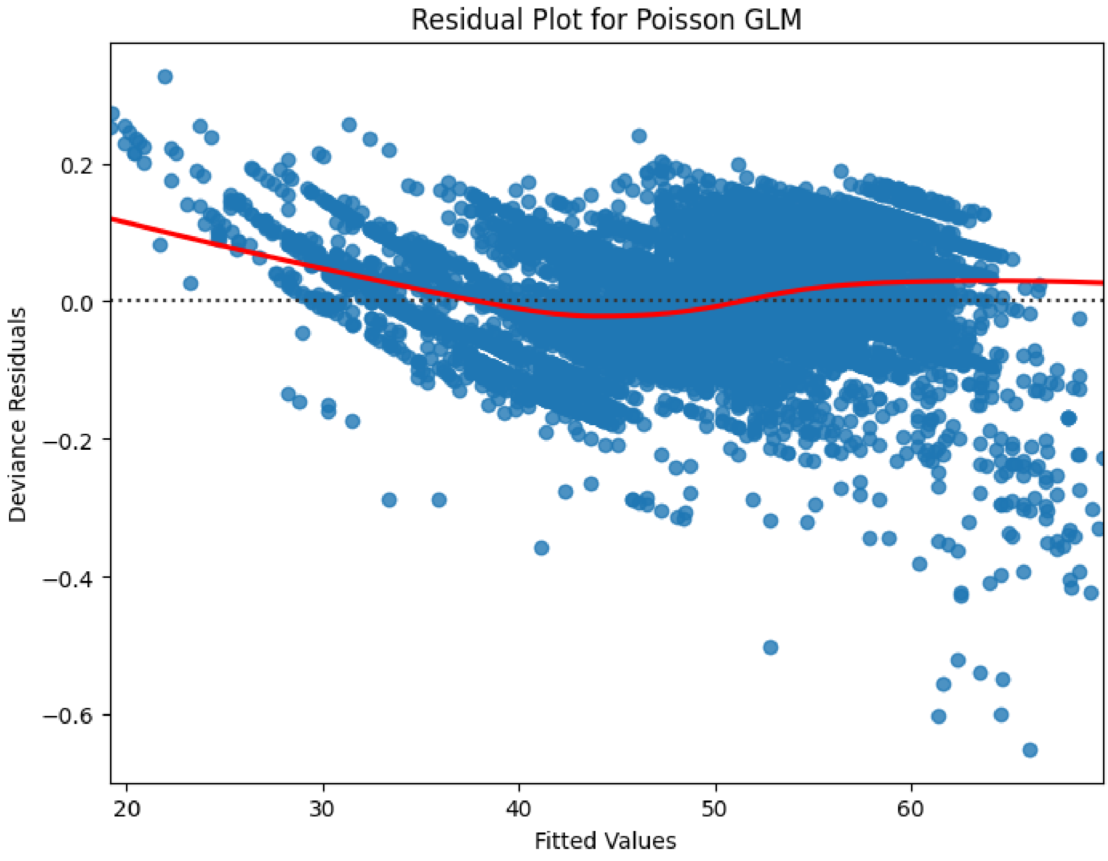
In [24]:

```
filtered_data2 = data[(data['mpg'] < 100) & (data['price'] < 90000)]

# Linear Regression with Log Transformed Price
model2 = sm.OLS.from_formula('np.log(price) ~ mpg', data=filtered_data2)
result2 = model2.fit()

plt.scatter(filtered_data2['mpg'], filtered_data2['price'])
plt.plot(filtered_data2['mpg'], np.exp(result2.fittedvalues), color='red')
plt.xlabel('MPG')
plt.ylabel('Price')
plt.title('Scatter Plot of MPG vs Price with Regression Line')
plt.show()
```



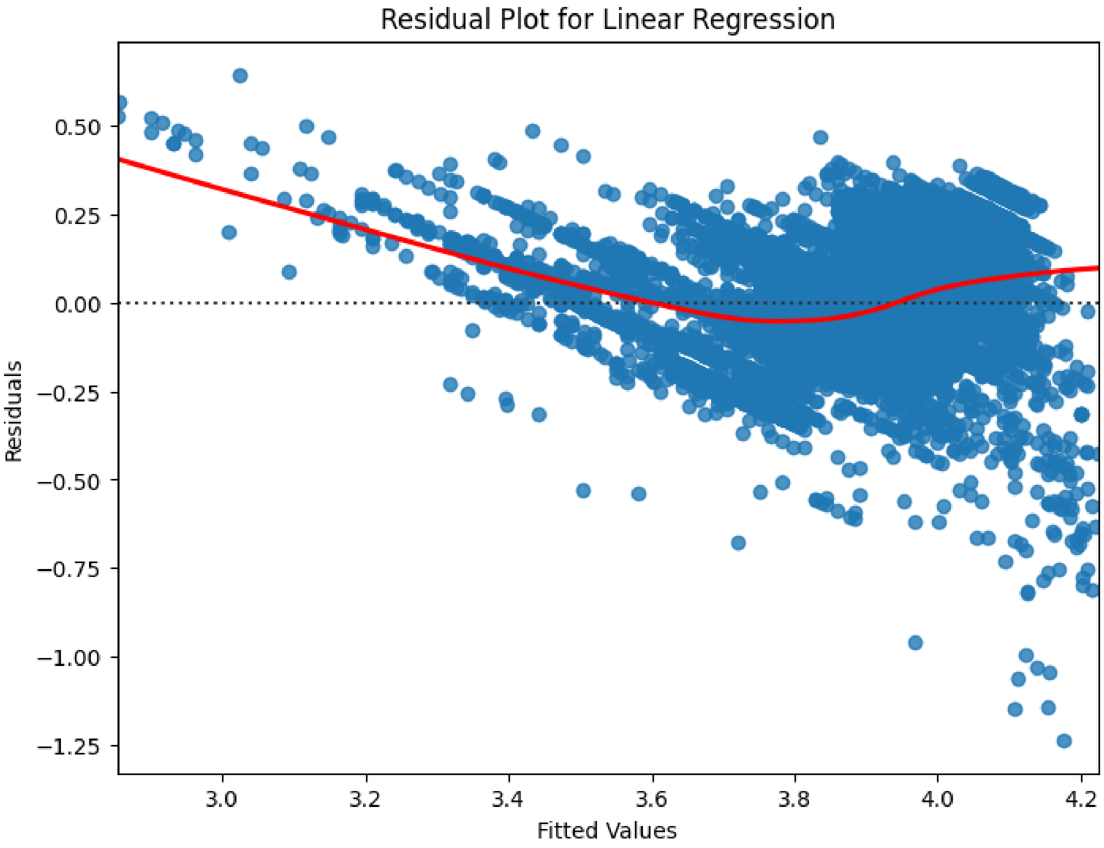
In [31]:

```
filtered_data2 = data[(data['mpg'] < 100) & (data['price'] < 90000)]

# GLM with Log Transformed MPG
model3 = sm.GLM.from_formula('np.log(mpg) ~ price', data=filtered_data2, f
result3 = model3.fit()

# Calculate Poisson Deviance Residuals
residuals = result3.resid_deviance

# Plot Residuals
fig, ax = plt.subplots(figsize=(8, 6))
sns.residplot(x=np.exp(result3.fittedvalues), y=residuals, lowess=True, li
plt.xlabel('Fitted Values')
plt.ylabel('Deviance Residuals')
plt.title('Residual Plot for Poisson GLM')
plt.show()
```



In [33]:

```
# Removing Outliers
filtered_data4 = filtered_data2[filtered_data2['price'] < 100000]

# Linear Regression with Log Transformed MPG
model4 = sm.OLS.from_formula('np.log(mpg) ~ price', data=filtered_data4)
result4 = model4.fit()

# Calculate Linear Regression Residuals
residuals_lr = result4.resid

# Plot Residuals
fig, ax = plt.subplots(figsize=(8, 6))
sns.residplot(x=result4.fittedvalues, y=residuals_lr, lowess=True, line_kw
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')
plt.title('Residual Plot for Linear Regression')
plt.show()
```



In [ ]: