CS6140: Machine Learning - Prof. Ahmad Uzair

Assignment 1: Data Analysis

In [553…
```
# importing the libraries
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
```

Reading the Data

In [554…
```
# read data from csv file
df = pd.read_csv("https://raw.githubusercontent.com/DrUzair/MLSD/master/Datasets/vehicle
```

```
C:\Users\bhara\AppData\Local\Temp\ipykernel_16932\287757298.py:2: DtypeWarning: Columns
(73,74,76) have mixed types. Specify dtype option on import or set low_memory=False.
  df = pd.read_csv("https://raw.githubusercontent.com/DrUzair/MLSD/master/Datasets/vehic
les.csv")
```

In [555…
```
df.shape
```

Out[555]:
```
(40081, 83)
```

In [556…
```
# print head of data frame with help of head function
df.head()
```

Out[556]:

	barrels08	barrelsA08	charge120	charge240	city08	city08U	cityA08	cityA08U	cityCD	cityE	…	mfrCode
0	15.695714	0.0	0.0	0.0	19	0.0	0	0.0	0.0	0.0	…	NaN
1	29.964545	0.0	0.0	0.0	9	0.0	0	0.0	0.0	0.0	…	NaN
2	12.207778	0.0	0.0	0.0	23	0.0	0	0.0	0.0	0.0	…	NaN
3	29.964545	0.0	0.0	0.0	10	0.0	0	0.0	0.0	0.0	…	NaN
4	17.347895	0.0	0.0	0.0	17	0.0	0	0.0	0.0	0.0	…	NaN

5 rows × 83 columns

In [557…
```
print(df.dtypes)
```

```
barrels08      float64
barrelsA08     float64
charge120      float64
charge240      float64
city08           int64
```

```
                ...
modifiedOn     object
startStop      object
phevCity        int64
phevHwy         int64
phevComb        int64
Length: 83, dtype: object
```

In [558…]
```
df.describe()
```

Out[558]:

	barrels08	barrelsA08	charge120	charge240	city08	city08U	cityA08	cityA0
count	40081.000000	40081.000000	40081.0	40081.000000	40081.000000	40081.000000	40081.000000	40081.000
mean	17.363564	0.220069	0.0	0.036086	18.213318	5.494777	0.616077	0.466
std	4.597119	1.143270	0.0	0.534894	7.397433	11.027993	4.739349	4.563
min	0.060000	0.000000	0.0	0.000000	6.000000	0.000000	0.000000	0.000
25%	14.330870	0.000000	0.0	0.000000	15.000000	0.000000	0.000000	0.000
50%	16.480500	0.000000	0.0	0.000000	17.000000	0.000000	0.000000	0.000
75%	19.388824	0.000000	0.0	0.000000	20.000000	12.273600	0.000000	0.000
max	47.087143	18.311667	0.0	12.000000	150.000000	150.000000	145.000000	145.083

8 rows × 59 columns

In [559…]
```
# identify rows with null or invalid value (NULL, Invalid, Empty, Unknown)
print(df.isnull().sum())
```

```
barrels08          0
barrelsA08         0
charge120          0
charge240          0
city08             0
                ...
modifiedOn         0
startStop      31704
phevCity           0
phevHwy            0
phevComb           0
Length: 83, dtype: int64
```

The Dependent Variable - UCity

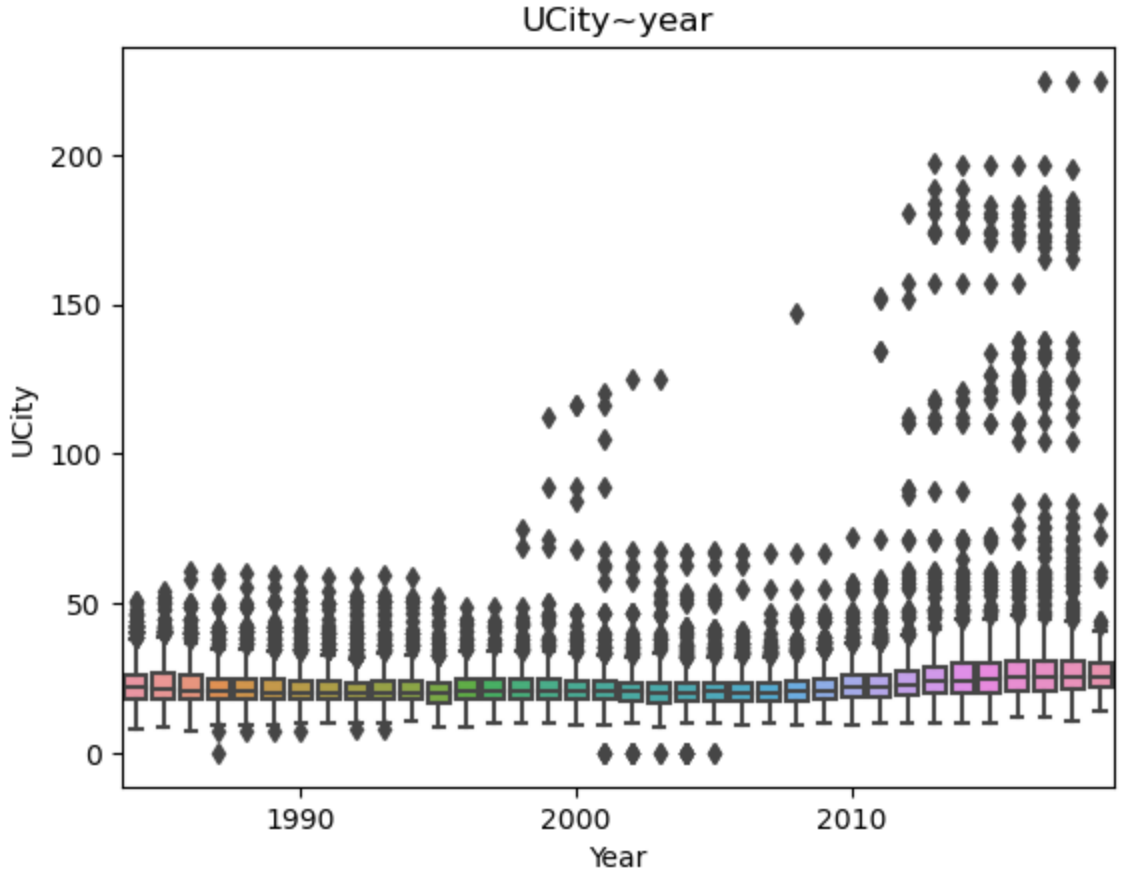
In [560…]
```
df['UCity'].describe()
```

Out[560]:
```
count    40081.000000
mean        22.981798
std         10.473444
min          0.000000
25%         18.110500
50%         21.296500
75%         25.700000
max        224.800000
Name: UCity, dtype: float64
```

In [561…]
```
# box plot UCity~year
ax = sns.boxplot(data = df, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "UCity~year", xlabel = "Year")
```

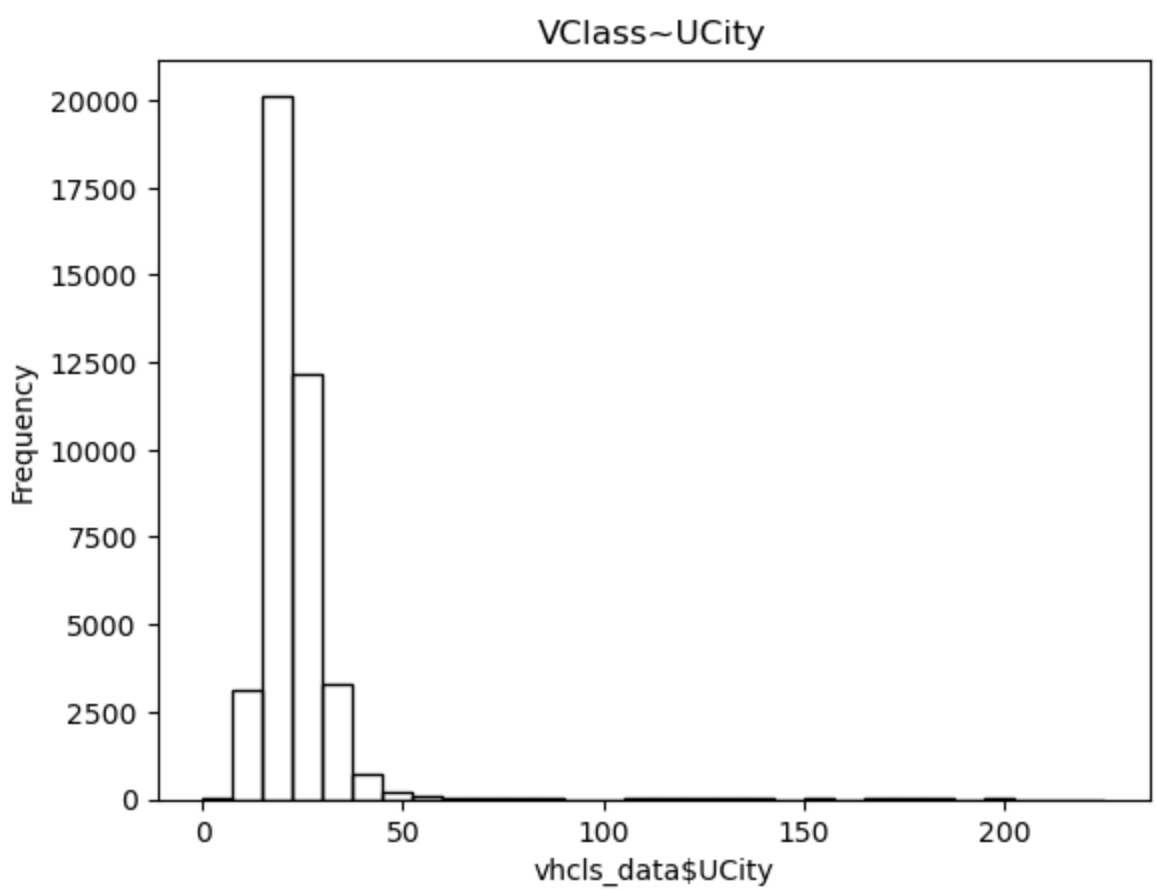
```
# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



In [562…

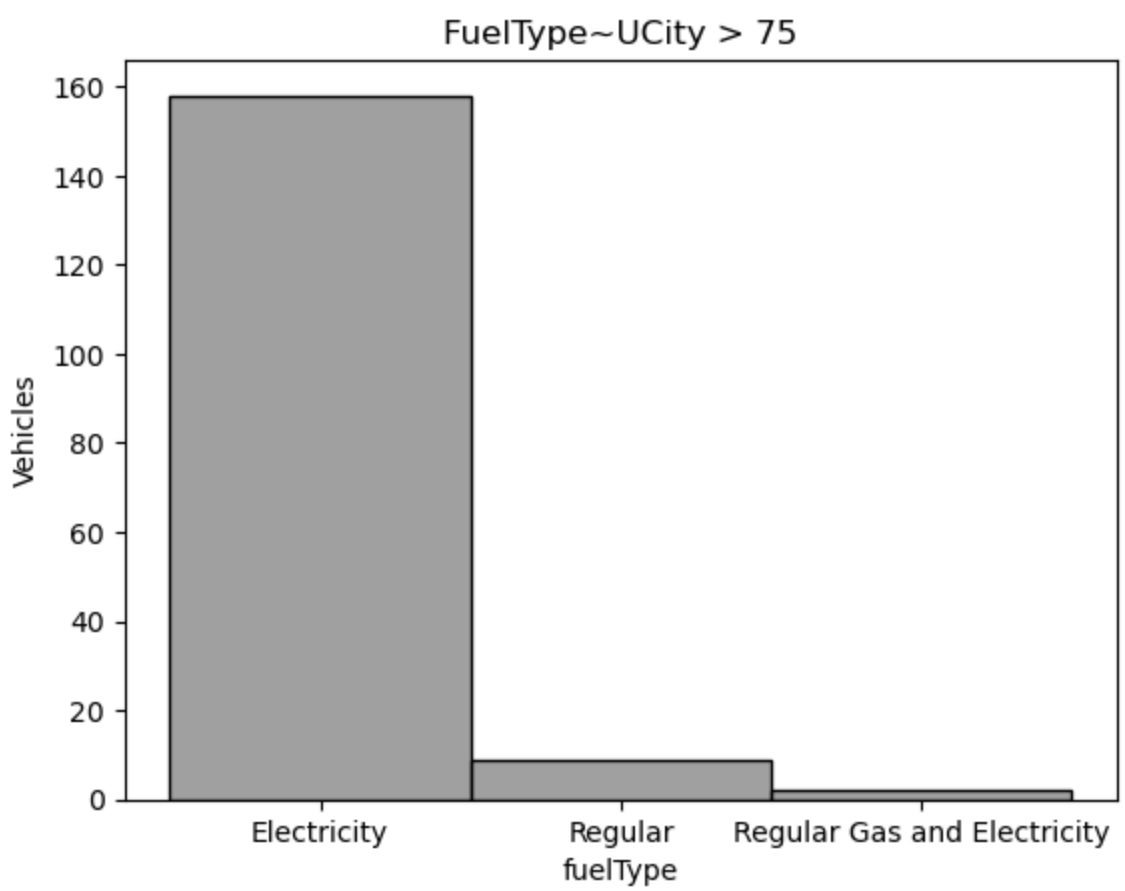
```
# plot histogram of vhcls_data$UCity
ax = sns.histplot(data = df['VClass'], x = df['UCity'],
                  bins = 30, color = 'none')

# set title and redefine the xlabel
ax.set(title = "VClass~UCity", xlabel = "vhcls_data$UCity",
       ylabel = "Frequency")

plt.show()
```



In [563…
```
# filter out UCity > 75
df_f = df.loc[df['UCity'] > 75]
df_f['fuelType']
```

Out[563]:
```
7138     Electricity
7139     Electricity
8143     Electricity
8144     Electricity
8147     Electricity
            ...
33032    Electricity
33373        Regular
33409    Electricity
33410    Electricity
33411    Electricity
Name: fuelType, Length: 169, dtype: object
```

In [564…
```
# plot histogram of fuelType~UCity
ax = sns.histplot(data = df.loc[df['UCity'] > 75],
                  x = df.loc[df['UCity'] > 75]['fuelType'],
                  color = 'grey', bins = 50)

# set title and redefine the xlabel
ax.set(title = "FuelType~UCity > 75", xlabel = "fuelType",
       ylabel = "Vehicles")

plt.show()
```



In [565…
```
df_zero_city = df.loc[df['UCity'] == 0]
df_zero_city[['make', 'model', 'fuelType', 'atvType']]
```

Out[565]:

	make	model	fuelType	atvType
8127	Ford	F150 Dual-fuel 2WD (CNG)	Gasoline or natural gas	Bifuel (CNG)
8128	Ford	F150 Dual-fuel 4WD (CNG)	Gasoline or natural gas	Bifuel (CNG)
8129	Ford	F150 Dual-fuel 2WD (LPG)	Gasoline or propane	Bifuel (LPG)
8130	Ford	F150 Dual-fuel 4WD (LPG)	Gasoline or propane	Bifuel (LPG)
9174	Dodge	Ram Van 2500 2WD CNG	CNG	CNG
9175	Dodge	Ram Wagon 2500 2WD CNG	CNG	CNG
9183	Ford	F150 Dual-fuel 2WD (CNG)	Gasoline or natural gas	Bifuel (CNG)
9184	Ford	F150 Dual-fuel 4WD (CNG)	Gasoline or natural gas	Bifuel (CNG)
9185	Ford	F150 Dual-fuel 2WD (LPG)	Gasoline or propane	Bifuel (LPG)
9186	Ford	F150 Dual-fuel 4WD (LPG)	Gasoline or propane	Bifuel (LPG)
10282	Ford	F150 Dual-fuel 2WD (LPG)	Gasoline or propane	Bifuel (LPG)
10283	Ford	F150 Dual-fuel 4WD (LPG)	Gasoline or propane	Bifuel (LPG)
11584	Ford	F150 Dual-fuel 2WD (LPG)	Gasoline or propane	Bifuel (LPG)
11585	Ford	F150 Dual-fuel 4WD (LPG)	Gasoline or propane	Bifuel (LPG)
11586	Chevrolet	Express Cargo (Bi-fuel)	Gasoline or natural gas	Bifuel (CNG)
11587	Chevrolet	Express Passenger (Bi-fuel)	Gasoline or natural gas	Bifuel (CNG)
11588	GMC	Savana (cargo) (Bi-fuel)	Gasoline or natural gas	Bifuel (CNG)
11589	GMC	Savana Passenger (Bi-fuel)	Gasoline or natural gas	Bifuel (CNG)

11591	Chevrolet	Express Cargo (dedicated CNG)	CNG	CNG
11592	Chevrolet	Express Passenger (dedicated CNG)	CNG	CNG
11593	GMC	Savana Cargo (dedicated CNG)	CNG	CNG
11594	GMC	Savana Passenger (dedicated CNG)	CNG	CNG
12814	Dodge	Caravan/Grand Caravan 2WD	Gasoline or E85	FFV
12815	Chrysler	Voyager/Town and Country 2WD	Gasoline or E85	FFV
21505	Porsche	924 S	Regular	NaN

The Independent Variables

Numeric - atvType

In [566…
```
# find NAs for atvType and fill with Not Available
df.fillna('Not Available', inplace=True)
```

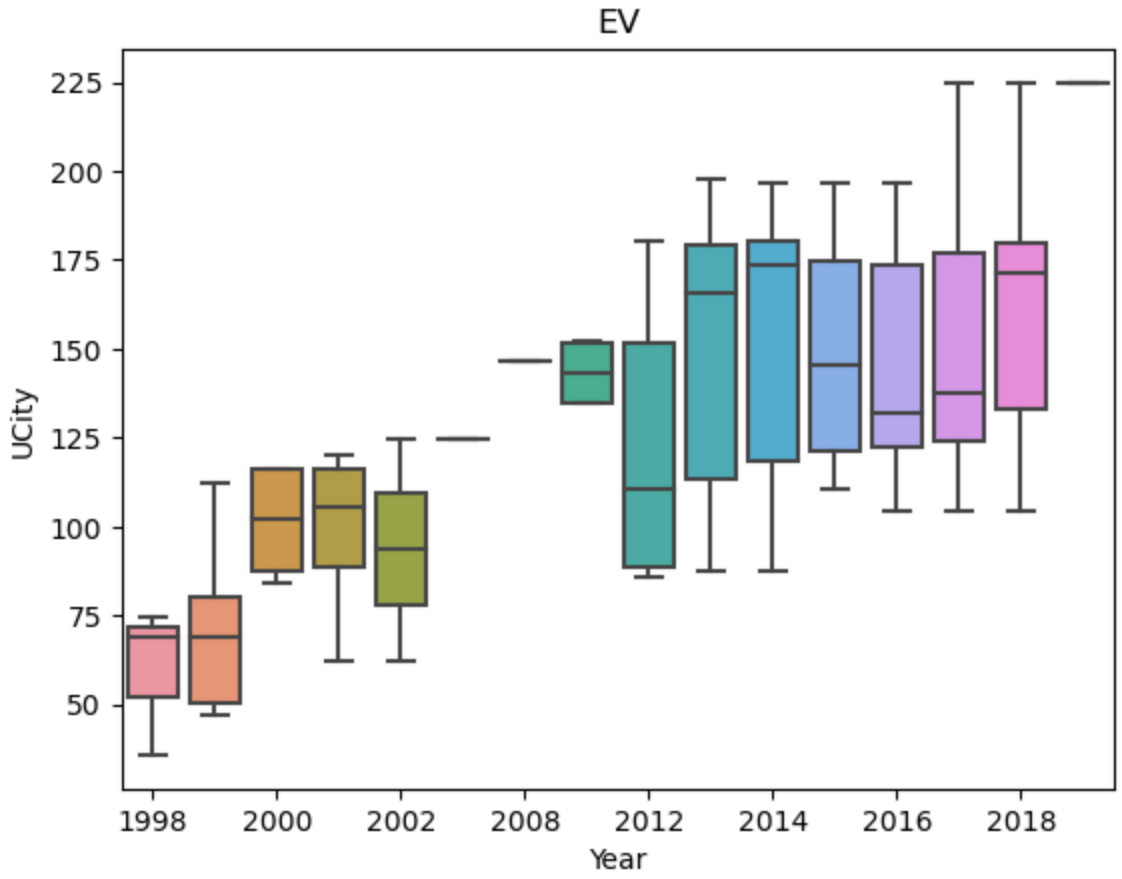
In [567…
```
# get count for unique types
df['atvType'].value_counts()
```

Out[567]:
```
Not Available     36707
FFV                1412
Diesel             1070
Hybrid              539
EV                  168
Plug-in Hybrid      107
CNG                  50
Bifuel (CNG)         20
Bifuel (LPG)          8
Name: atvType, dtype: int64
```

In [568…
```
# filter atvs for not available type
df_atv = df[df['atvType'] != 'Not Available']
```

In [569…
```
# box plot atvType~year = EV

df_ev = df_atv[df_atv['atvType'] == 'EV']
ax = sns.boxplot(data = df_ev, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "EV", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 2) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

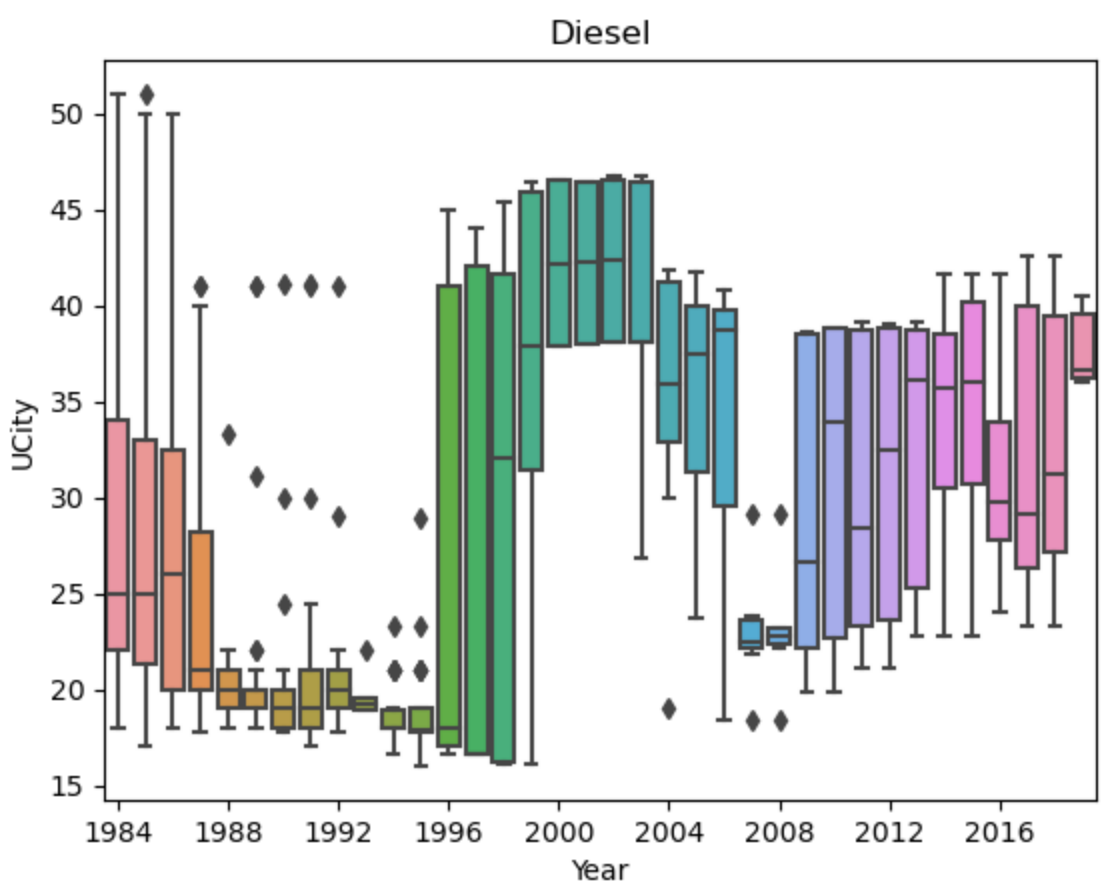
```
x_ticklabels: ['1998', '1999', '2000', '2001', '2002', '2003', '2008', '2011', '2012',
'2013', '2014', '2015', '2016', '2017', '2018', '2019']
```



In [570…

```
# box plot atvType~year = Diesel

df_ev = df_atv[df_atv['atvType'] == 'Diesel']
ax = sns.boxplot(data = df_ev, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "Diesel", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 4) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

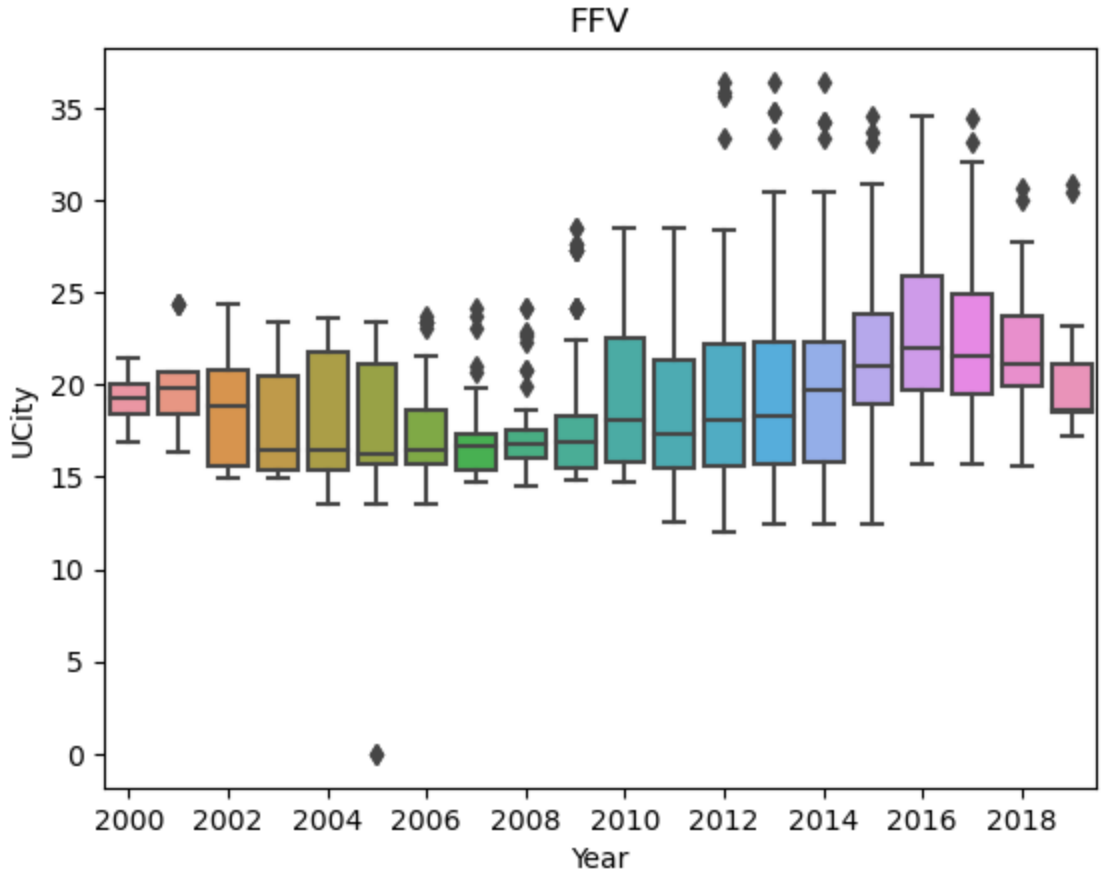
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



In [571…

```
# box plot atvType~year = FFV

df_ev = df_atv[df_atv['atvType'] == 'FFV']
ax = sns.boxplot(data = df_ev, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "FFV", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 2) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

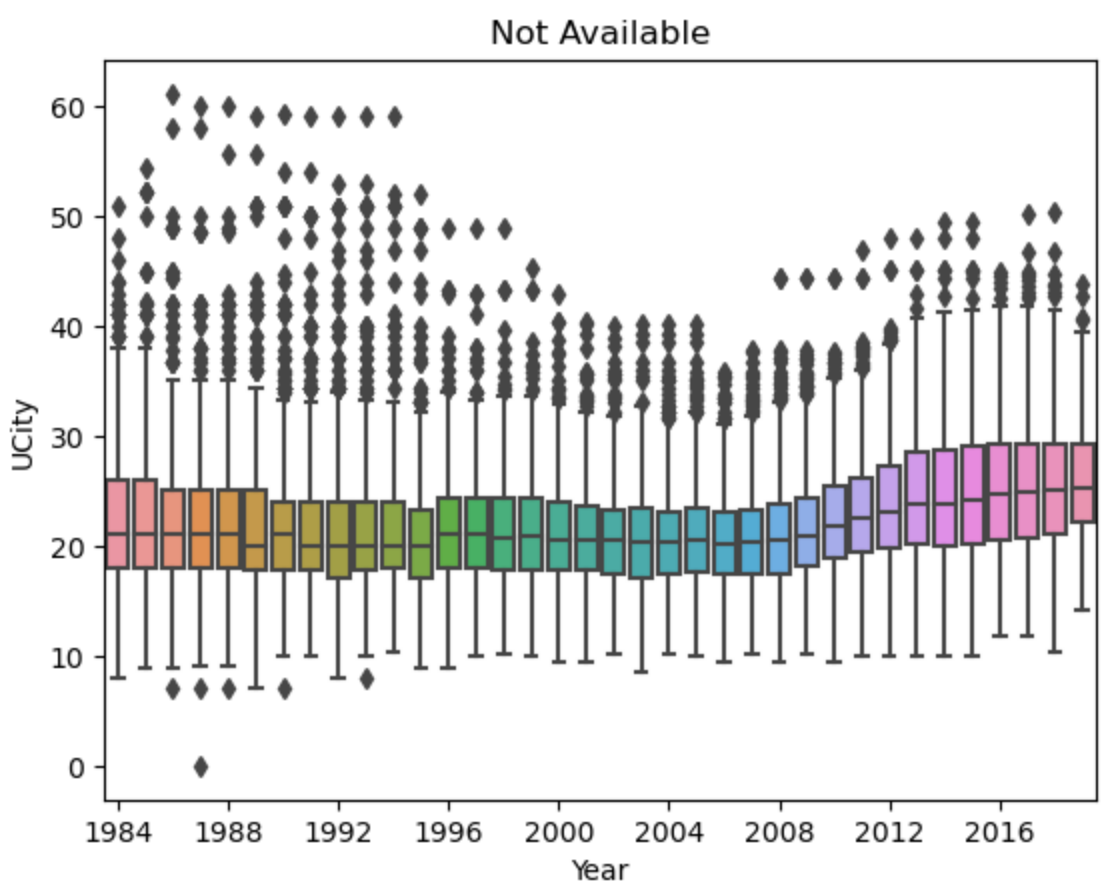
```
x_ticklabels: ['2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007', '2008',
'2009', '2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018', '2019']
```



In [572…

```
# box plot atvType~year = Not Available

df_na = df[df['atvType'] == 'Not Available']
ax = sns.boxplot(data = df_na, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "Not Available", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 4) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

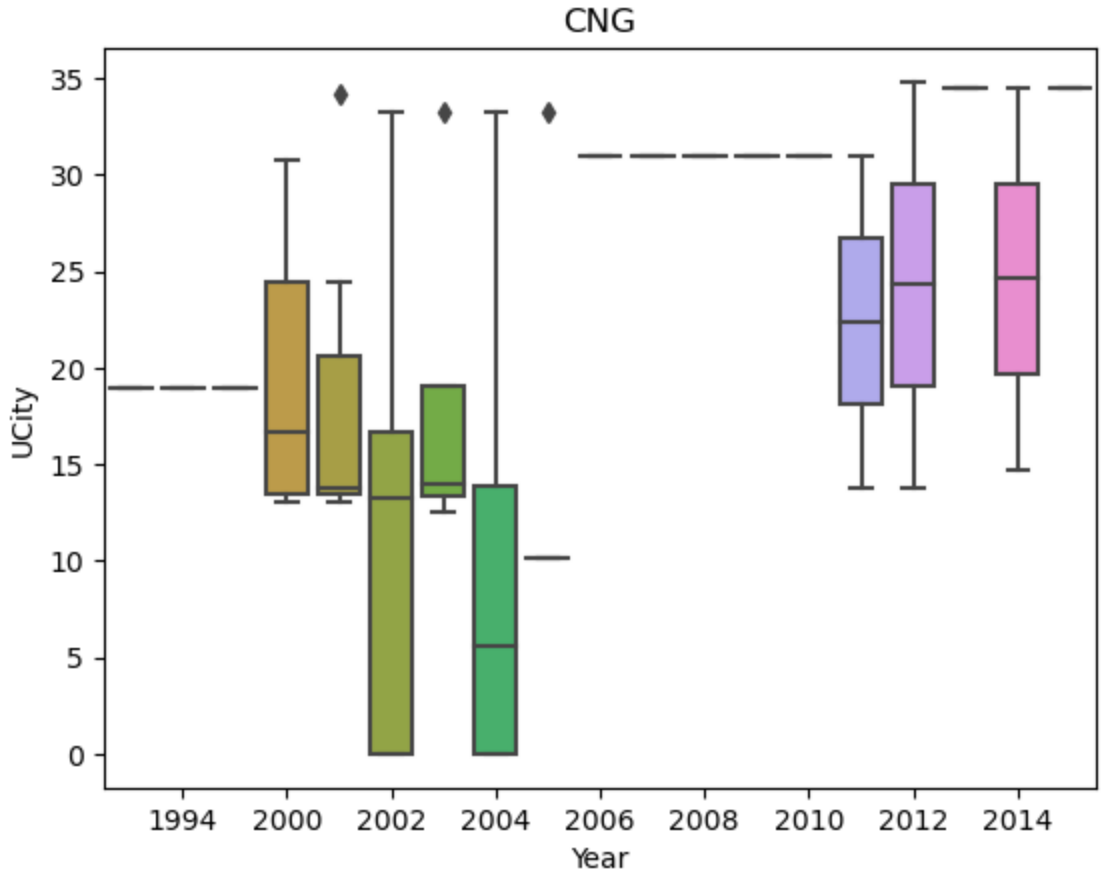
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



In [573…

```
# box plot atvType~year = CNG

df_ev = df_atv[df_atv['atvType'] == 'CNG']
ax = sns.boxplot(data = df_ev, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "CNG", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 2) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

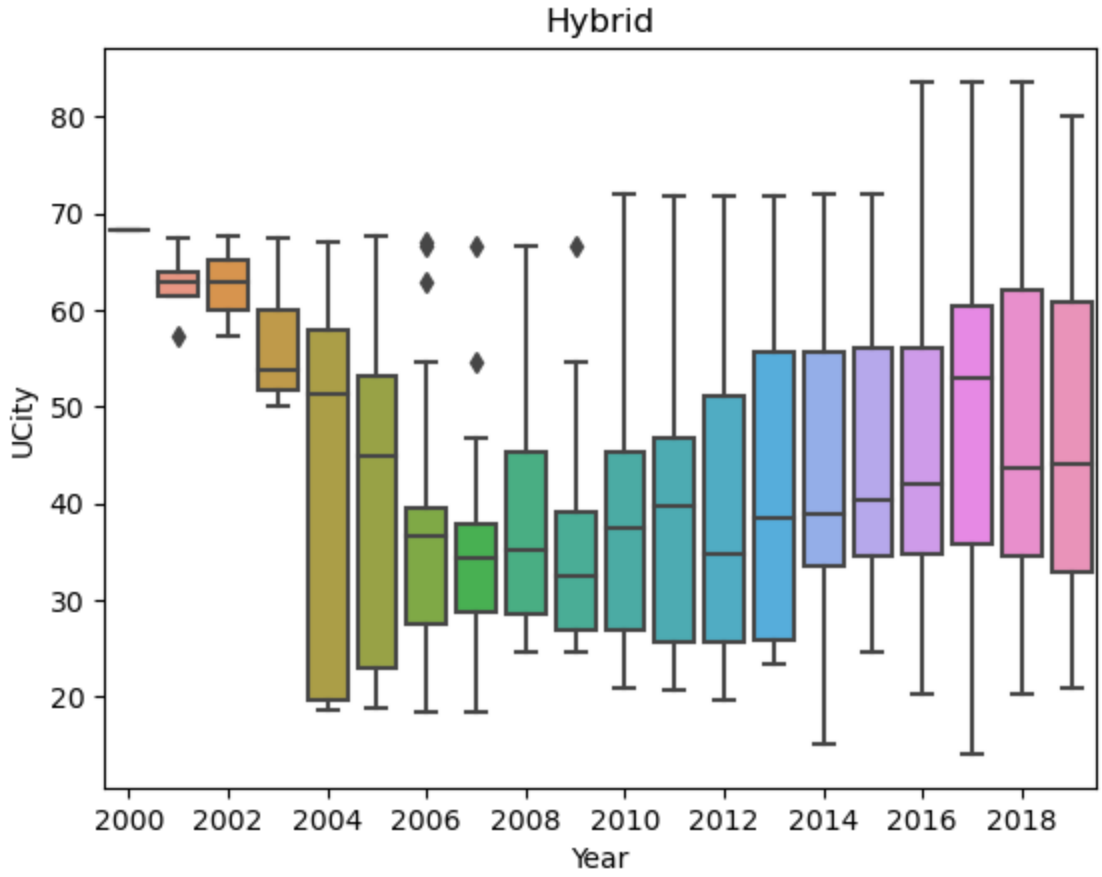
```
x_ticklabels: ['1993', '1994', '1995', '2000', '2001', '2002', '2003', '2004', '2005',
'2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015']
```



In [574…

```
# box plot atvType~year = Hybrid

df_ev = df_atv[df_atv['atvType'] == 'Hybrid']
ax = sns.boxplot(data = df_ev, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "Hybrid", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 2) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['2000', '2001', '2002', '2003', '2004', '2005', '2006', '2007', '2008',
'2009', '2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018', '2019']
```



Numeric - range

In [575…
```
df['range'].describe()
```

Out[575]:
```
count    40081.000000
mean         0.616377
std         11.133278
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max        335.000000
Name: range, dtype: float64
```

In [576…
```
df['range'].isnull()
```

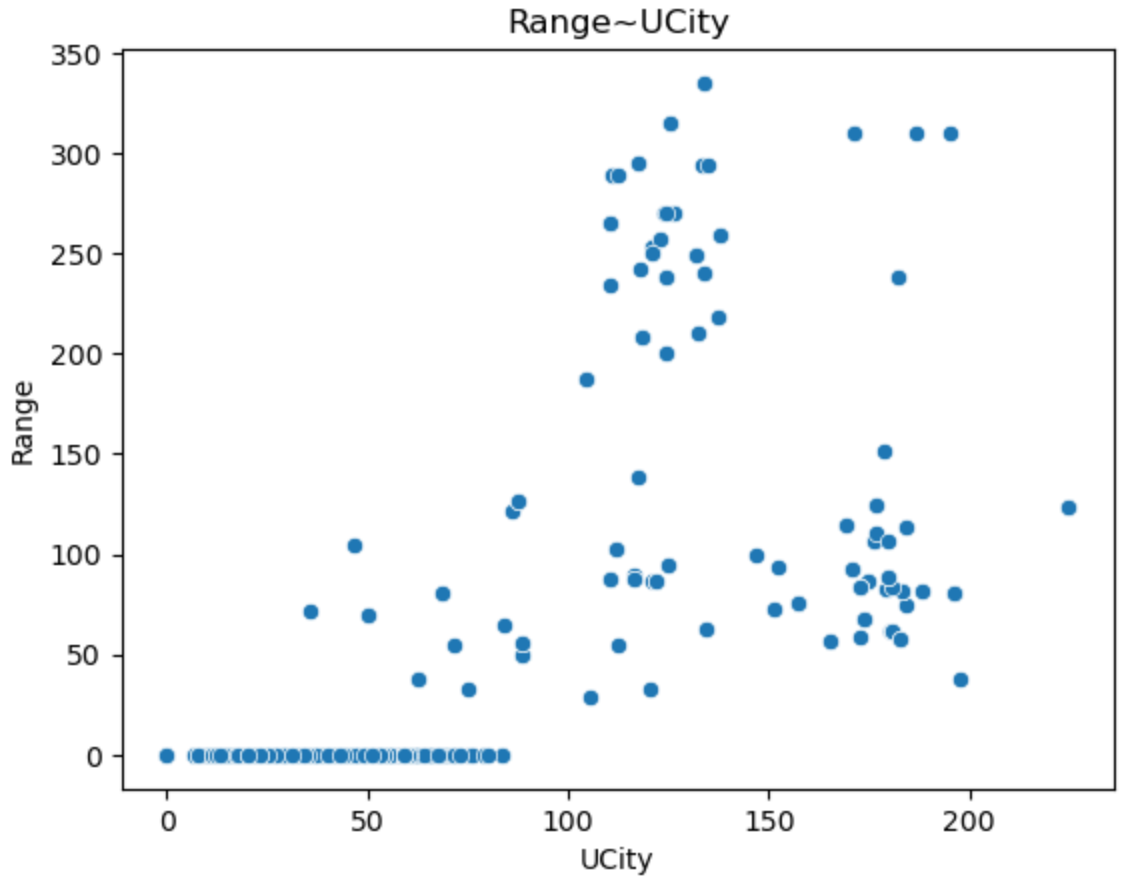
Out[576]:
```
0        False
1        False
2        False
3        False
4        False
         ...
40076    False
40077    False
40078    False
40079    False
40080    False
Name: range, Length: 40081, dtype: bool
```

In [577…
```
# filter out vehicles with no range information
df_range = df.loc[df['range'] != 0]

print(len(df_range['range']))
```

```
# correlation between UCity and range
print("Correlation between range and UCity: ", df['range'].corr(df['UCity']))
```

```
168
Correlation between range and UCity:  0.6013585777904693
```

In [578…
```
# plot scatter graph to compare range and UCity
sns.scatterplot(x = df['UCity'], y = df['range'])
plt.xlabel('UCity')
plt.ylabel('Range')
plt.title('Range~UCity')
plt.show()
```



Numeric - youSaveSpend

In [579…
```
df['youSaveSpend'].describe()
```

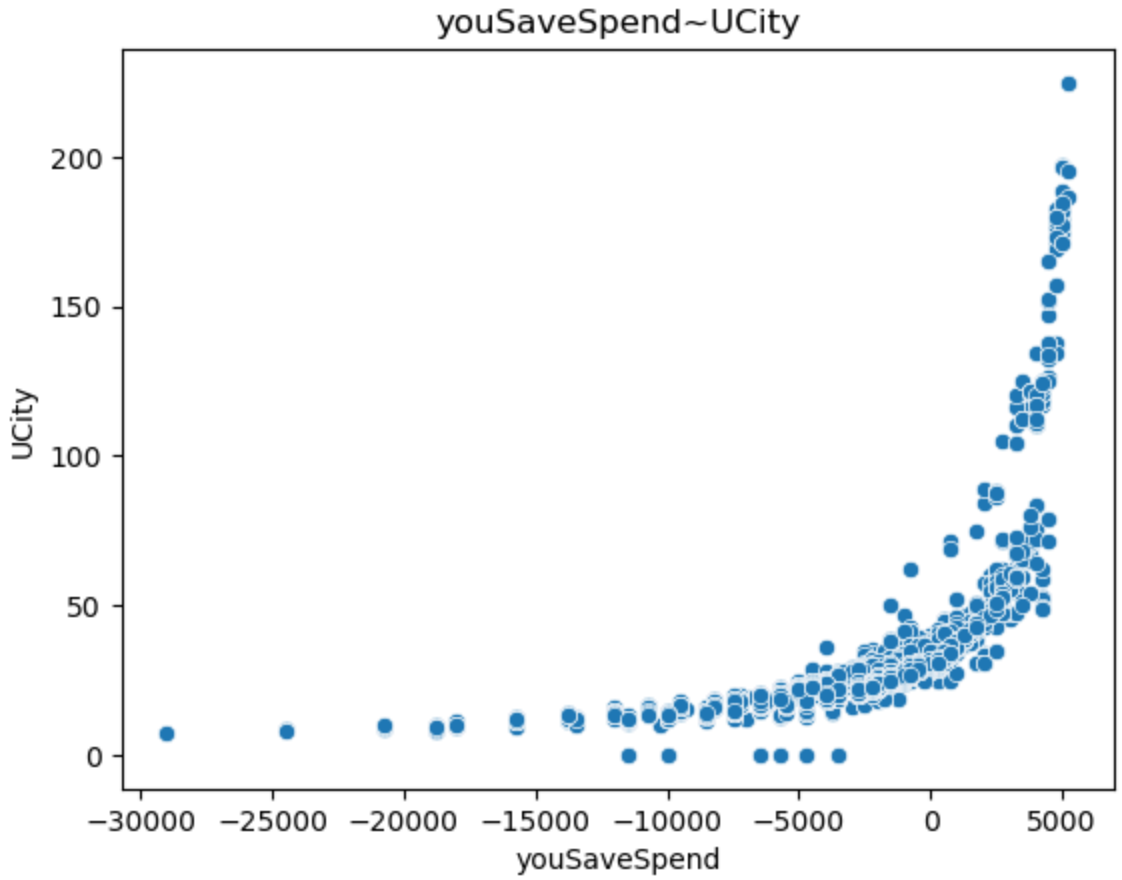
Out[579]:
```
count    40081.000000
mean     -4134.565006
std       3256.499139
min     -29000.000000
25%      -5750.000000
50%      -4000.000000
75%      -2000.000000
max       5250.000000
Name: youSaveSpend, dtype: float64
```

In [580…
```
# correlation between UCity and youSaveSpend
print("Correlation between youSaveSpend and UCity: ", df['youSaveSpend'].corr(df['UCity'
```

```
Correlation between youSaveSpend and UCity:  0.6583710195084375
```

In [581…
```
# plot scatter graph to compare youSaveSpend and UCity
sns.scatterplot(x = df['youSaveSpend'], y = df['UCity'])
plt.xlabel('youSaveSpend')
plt.ylabel('UCity')
```

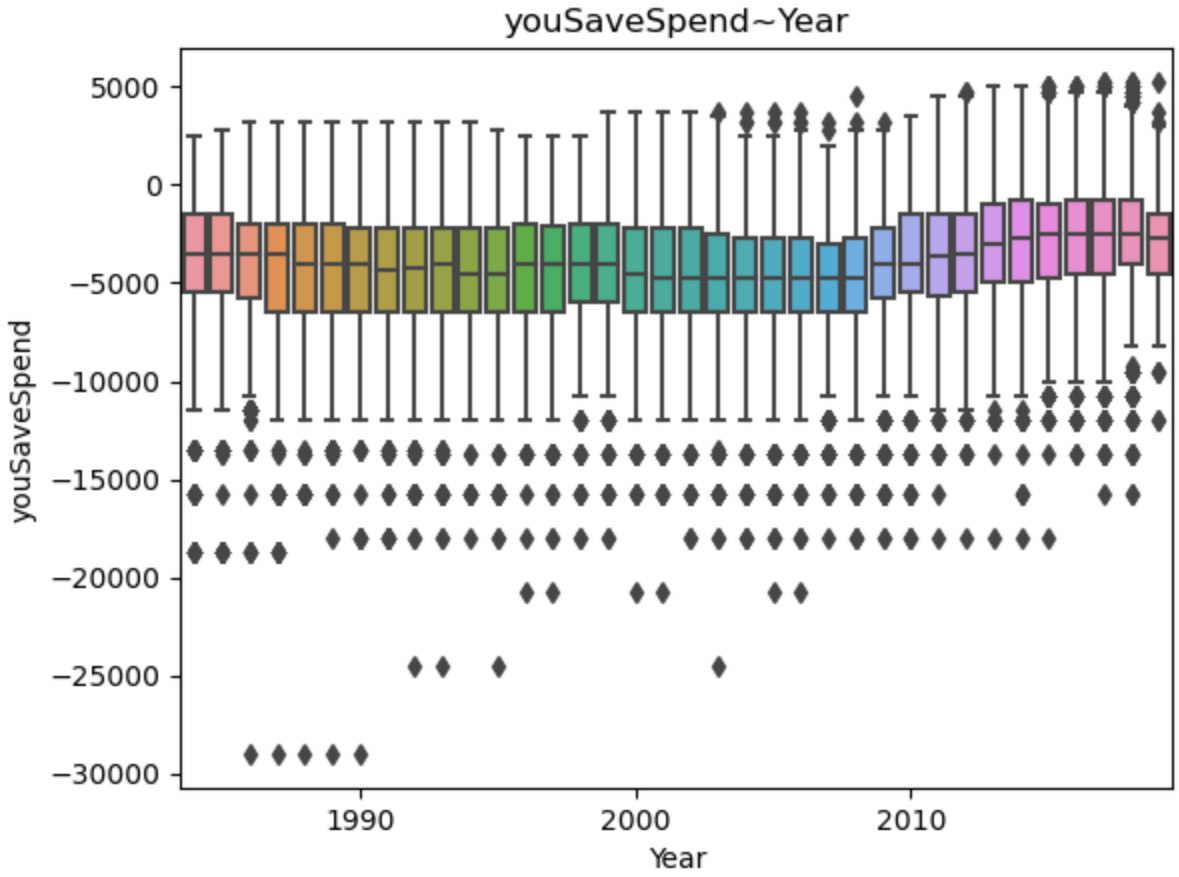
```
plt.title('youSaveSpend~UCity')
plt.show()
```



In [582…

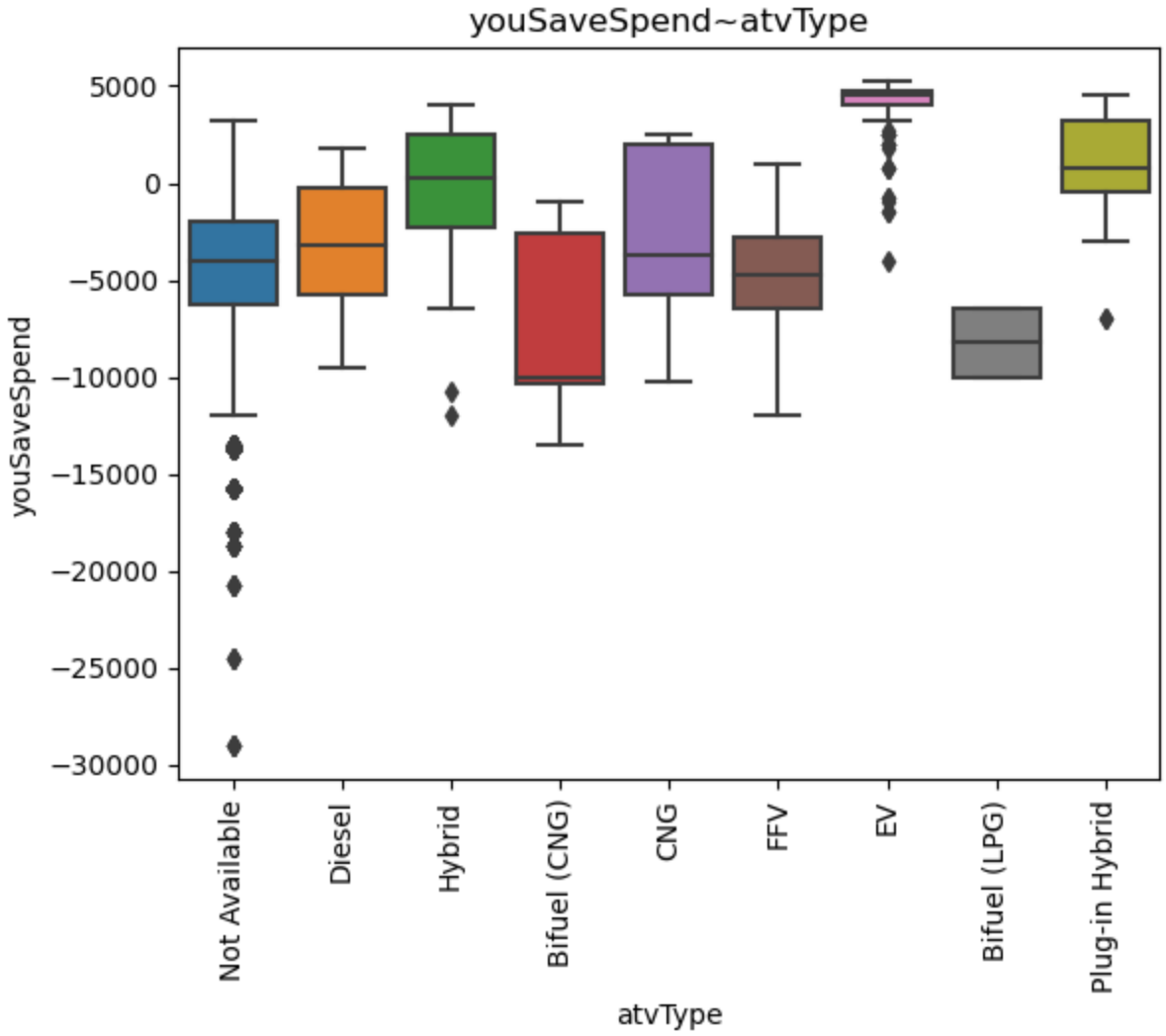
```
# box plot youSaveSpend~year

ax = sns.boxplot(data = df, x = "year", y = "youSaveSpend")

# set title and redefine the xlabel
ax.set(title = "youSaveSpend~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



In [583…

```
# box plot youSaveSpend~atvType

ax = sns.boxplot(data = df, x = "atvType", y = "youSaveSpend")

# set title and redefine the xlabel
ax.set(title = "youSaveSpend~atvType", xlabel = "atvType")

# get xtick labels
labels = ax.get_xticklabels()

# rotate labels by 90 degree
ax.set_xticklabels(labels = labels, rotation = 90)
plt.show()
```



Numeric - cylinders

In [584…
```
df['cylinders'].describe()
```

Out[584]:
```
count     40081.0
unique       10.0
top           4.0
freq      15475.0
Name: cylinders, dtype: float64
```

In [585…
```
df['cylinders'].unique()
```

Out[585]:
```
array([4.0, 12.0, 8.0, 6.0, 5.0, 10.0, 2.0, 3.0, 'Not Available', 16.0],
      dtype=object)
```

In [586…
```
df['cylinders'].replace('Not Available', 'NA\'s', inplace=True)
```

In [587…
```
df_c = df
df_c['cylinders'] = pd.to_numeric(df_c['cylinders'],
                                  errors = 'coerce')
```

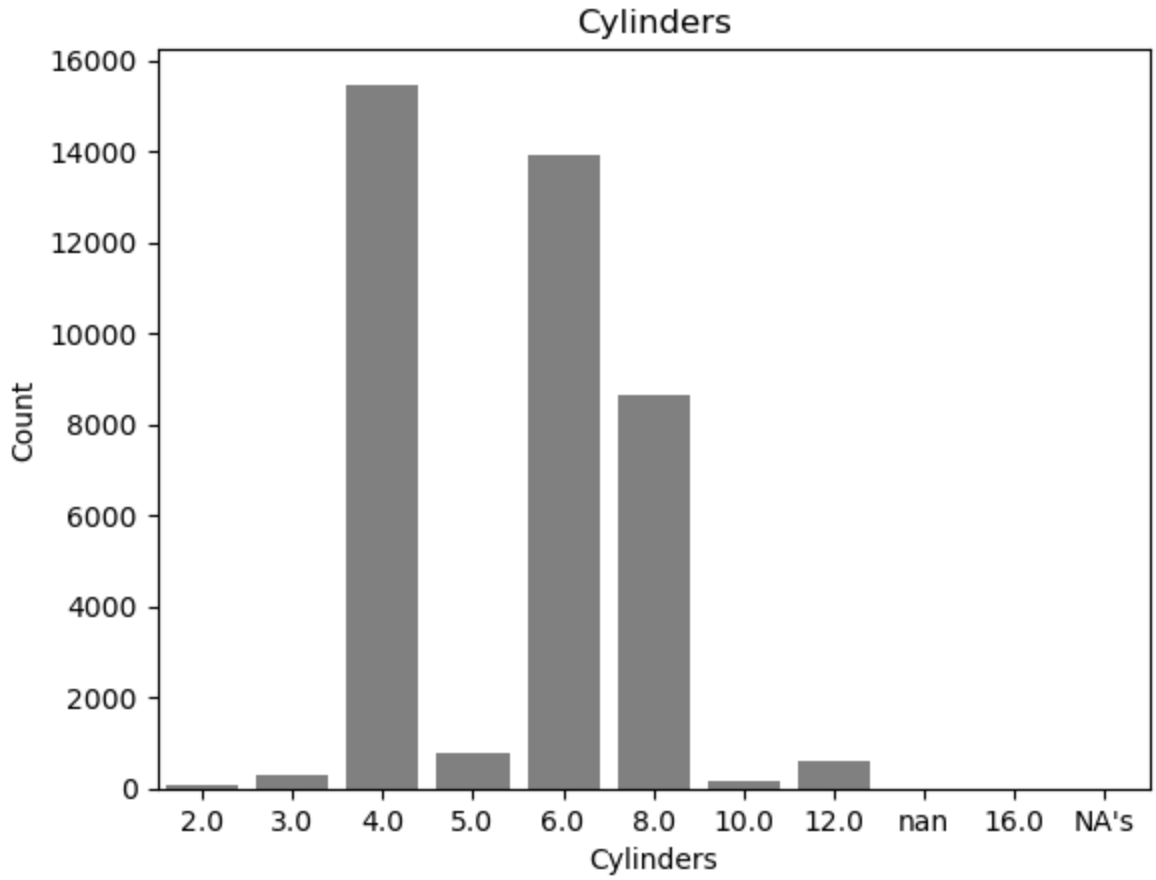
In [588…
```
# correlation between UCity and cylinders
print("Correlation between cylinders and UCity: ", df_c['cylinders'].corr(df_c['UCity']))
```

```
Correlation between cylinders and UCity:  -0.679927305583387
```

In [589…
```
# plot count graph for cylinders

# sort labels for plotting
```

```
cylinders = []
for val in df['cylinders'].unique():
    if isinstance(val, float):
        cylinders.append(val)

cylinders = sorted(cylinders)
cylinders.append("NA's")

ax = sns.countplot(x = df['cylinders'], color = 'grey', order = cylinders)
ax.set(title = "Cylinders", xlabel = "Cylinders", ylabel = "Count")
```

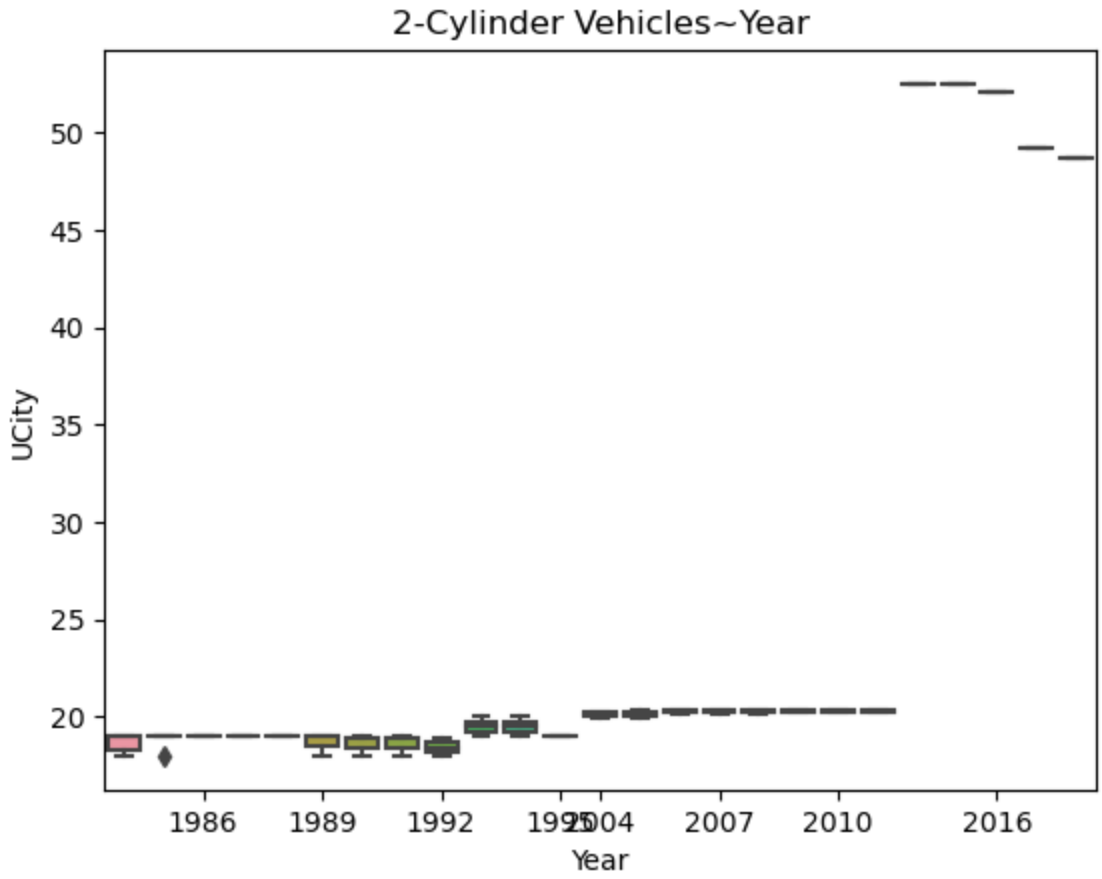
Out[589]:
```
[Text(0.5, 1.0, 'Cylinders'), Text(0.5, 0, 'Cylinders'), Text(0, 0.5, 'Count')]
```



In [590…
```
# box plot 2 cylinders vehicles
df_2c = df[df['cylinders'] == 2.0]

ax = sns.boxplot(data = df_2c, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "2-Cylinder Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 3) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
```

```
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

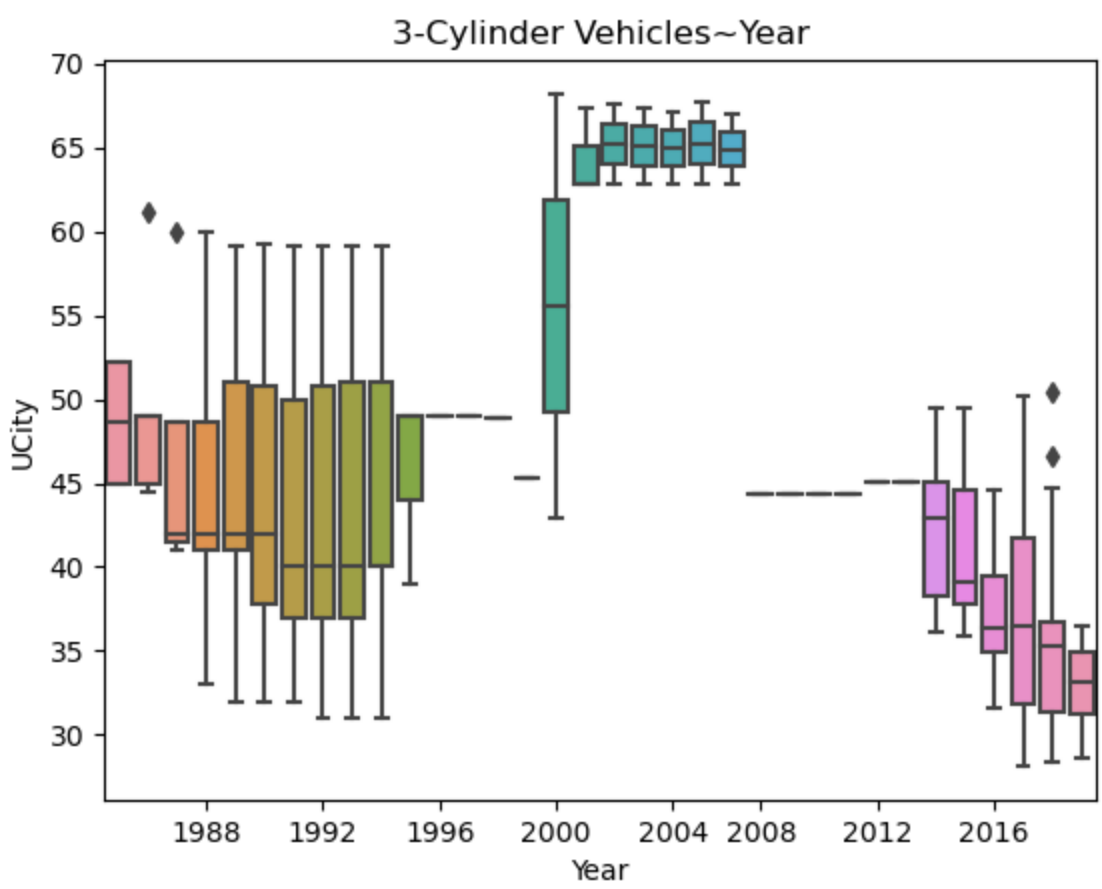
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011',
'2014', '2015', '2016', '2017', '2018']
```



In [591…

```
# box plot 3 cylinders vehicles
df_3c = df[df['cylinders'] == 3.0]

ax = sns.boxplot(data = df_3c, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "3-Cylinder Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 4) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

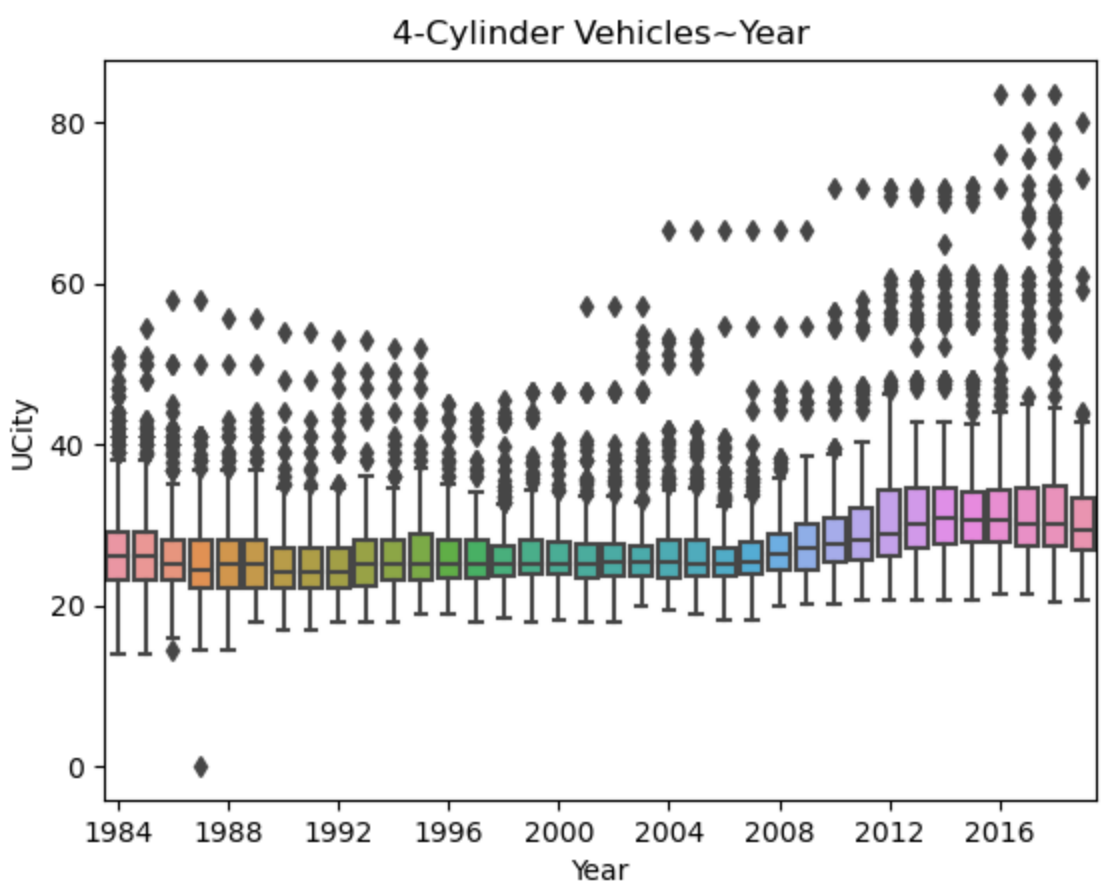
```
x_ticklabels: ['1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992', '1993',
'1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004',
'2005', '2006', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015', '2016',
'2017', '2018', '2019']
```



In [592…

```
# box plot 4 cylinders vehicles
df_4c = df[df['cylinders'] == 4.0]

ax = sns.boxplot(data = df_4c, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "4-Cylinder Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 4) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

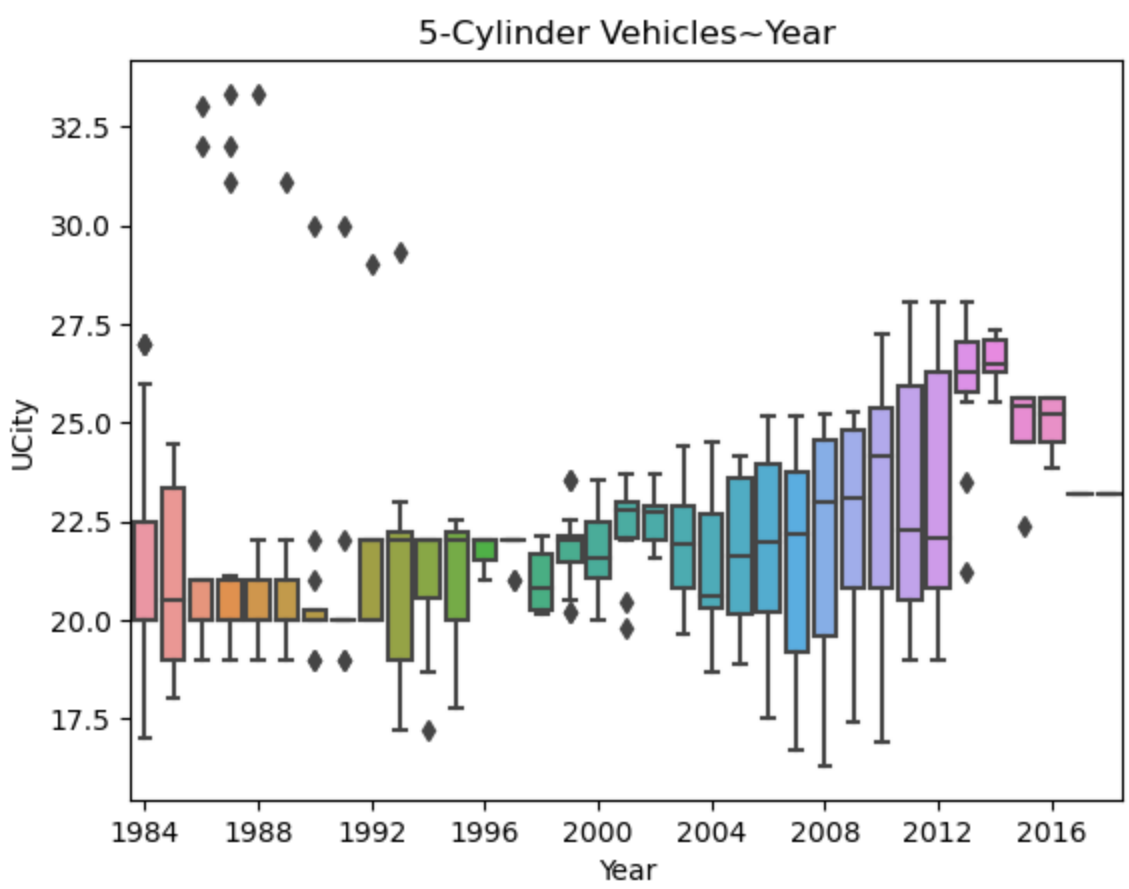
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



In [593…

```
# box plot 5 cylinders vehicles
df_5c = df[df['cylinders'] == 5.0]

ax = sns.boxplot(data = df_5c, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "5-Cylinder Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 4) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

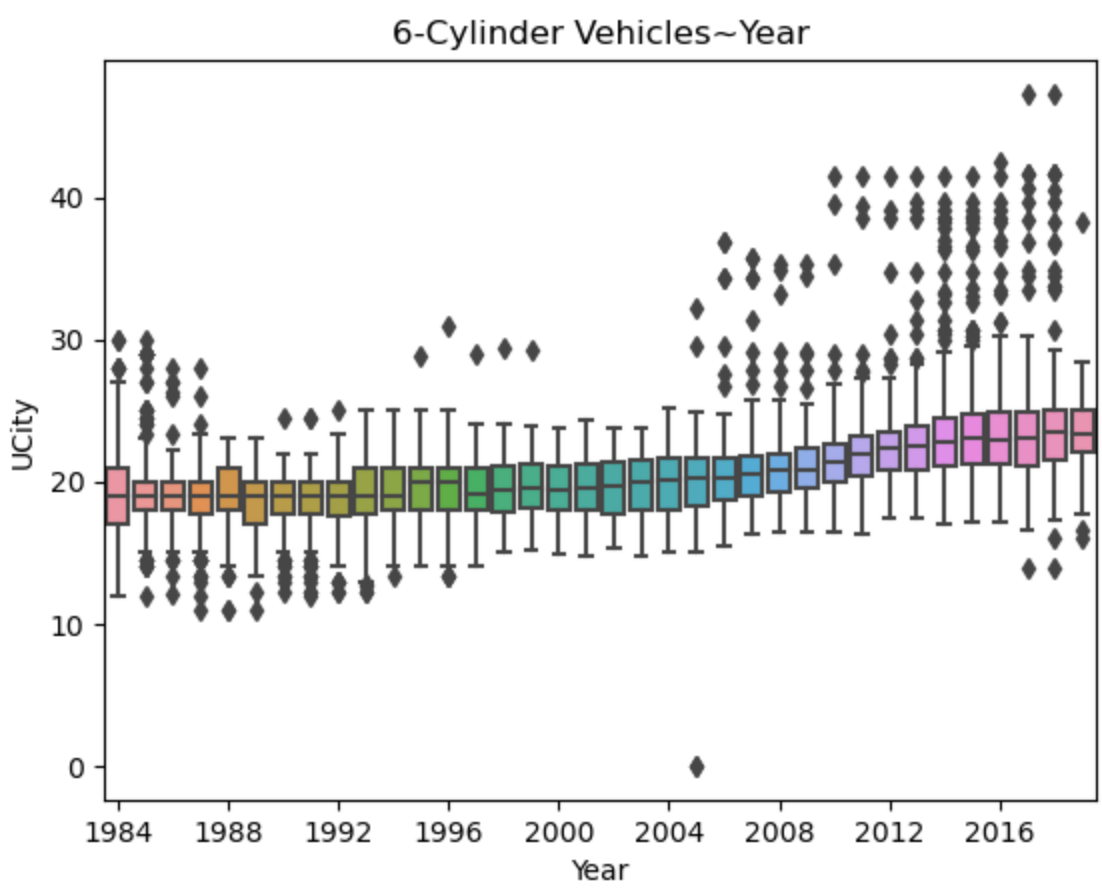
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018']
```



In [594…

```
# box plot 6 cylinders vehicles
df_6c = df[df['cylinders'] == 6.0]

ax = sns.boxplot(data = df_6c, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "6-Cylinder Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 4) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

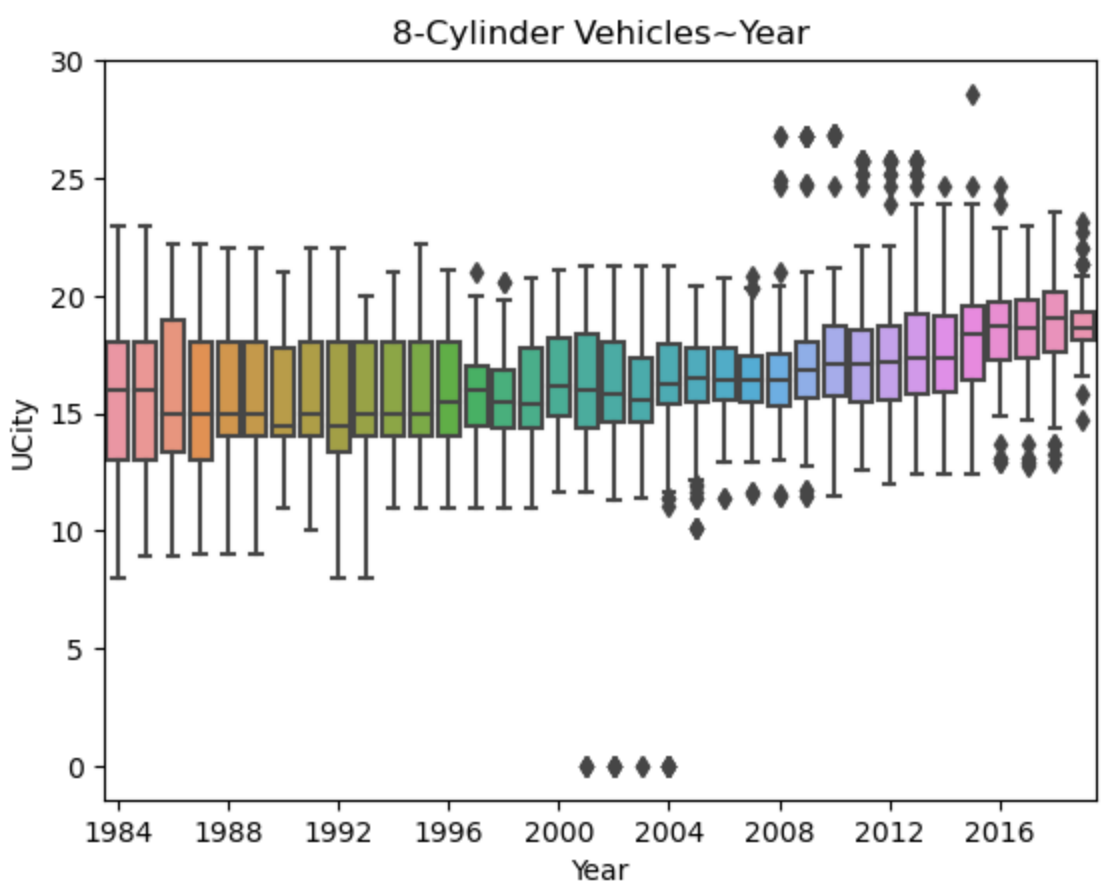
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



In [595…

```
# box plot 8 cylinders vehicles
df_8c = df[df['cylinders'] == 8.0]

ax = sns.boxplot(data = df_8c, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "8-Cylinder Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 4) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

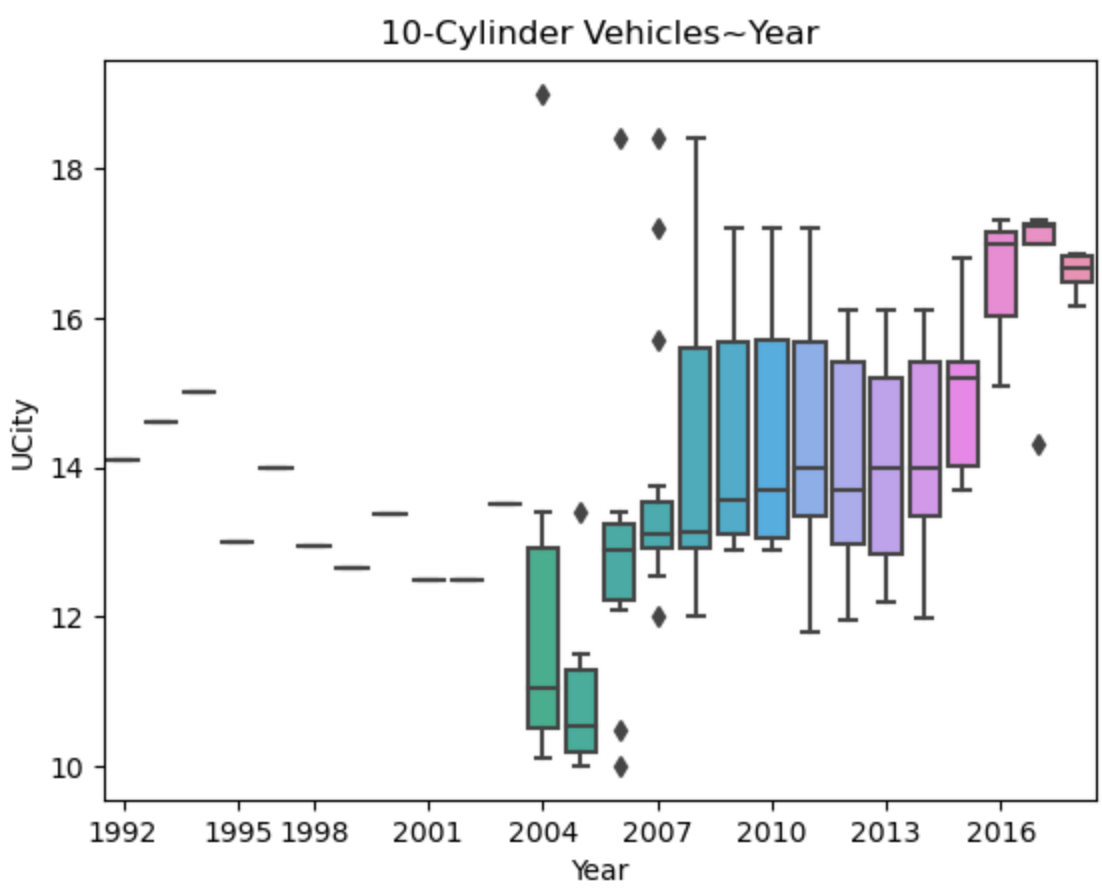
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



In [596…

```
# box plot 10 cylinders vehicles
df_10c = df[df['cylinders'] == 10.0]

ax = sns.boxplot(data = df_10c, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "10-Cylinder Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 3) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

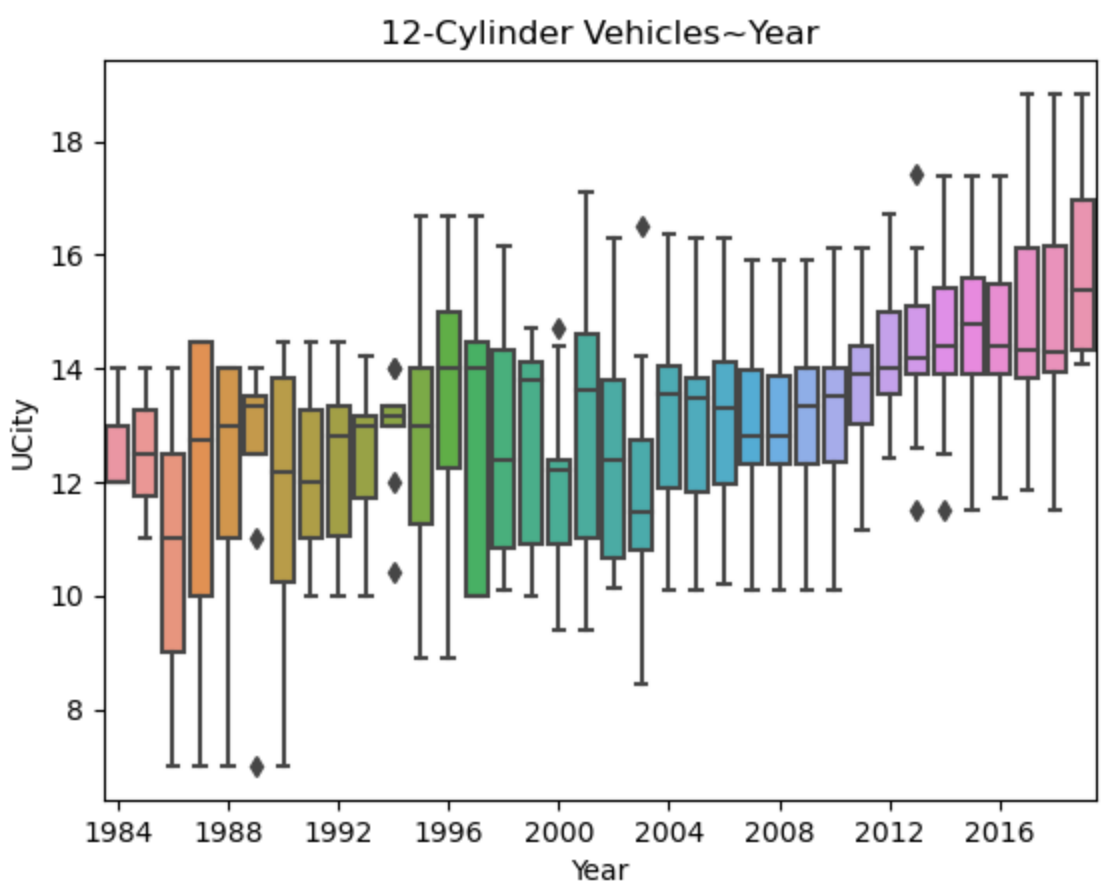
```
x_ticklabels: ['1992', '1993', '1994', '1995', '1996', '1998', '1999', '2000', '2001',
'2002', '2003', '2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012',
'2013', '2014', '2015', '2016', '2017', '2018']
```



In [597…

```
# box plot 12 cylinders vehicles
df_12c = df[df['cylinders'] == 12.0]

ax = sns.boxplot(data = df_12c, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "12-Cylinder Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 4) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

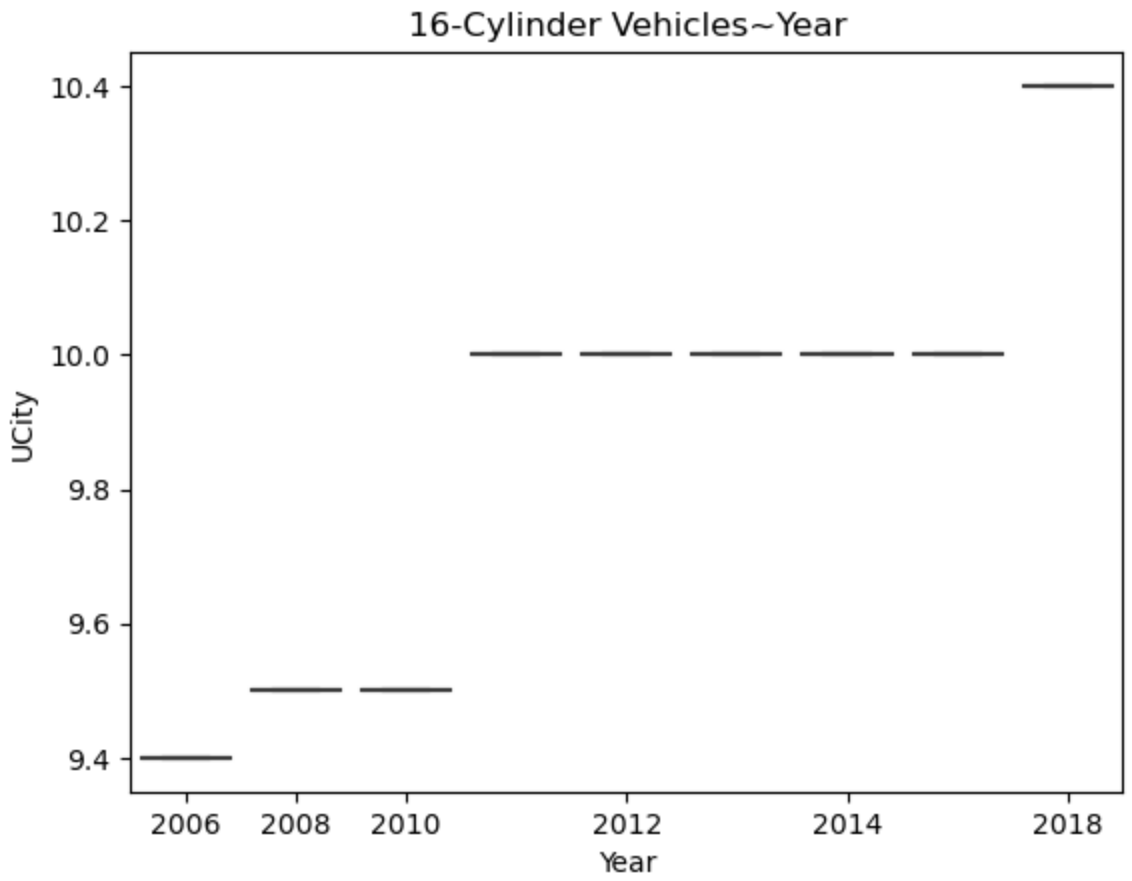
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



In [598…

```
# box plot 16 cylinders vehicles
df_16c = df[df['cylinders'] == 16.0]

ax = sns.boxplot(data = df_16c, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "16-Cylinder Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 2) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['2006', '2008', '2010', '2011', '2012', '2013', '2014', '2015', '2018']
```



Numeric - displ

In [599…
```
df['displ'].describe()
```

Out[599]:
```
count     40081.0
unique       67.0
top           2.0
freq       4063.0
Name: displ, dtype: float64
```

In [600…
```
# filter out Not Available displ
df_displ = df['displ'].replace('Not Available', np.NaN)
df_displ.dropna()
```

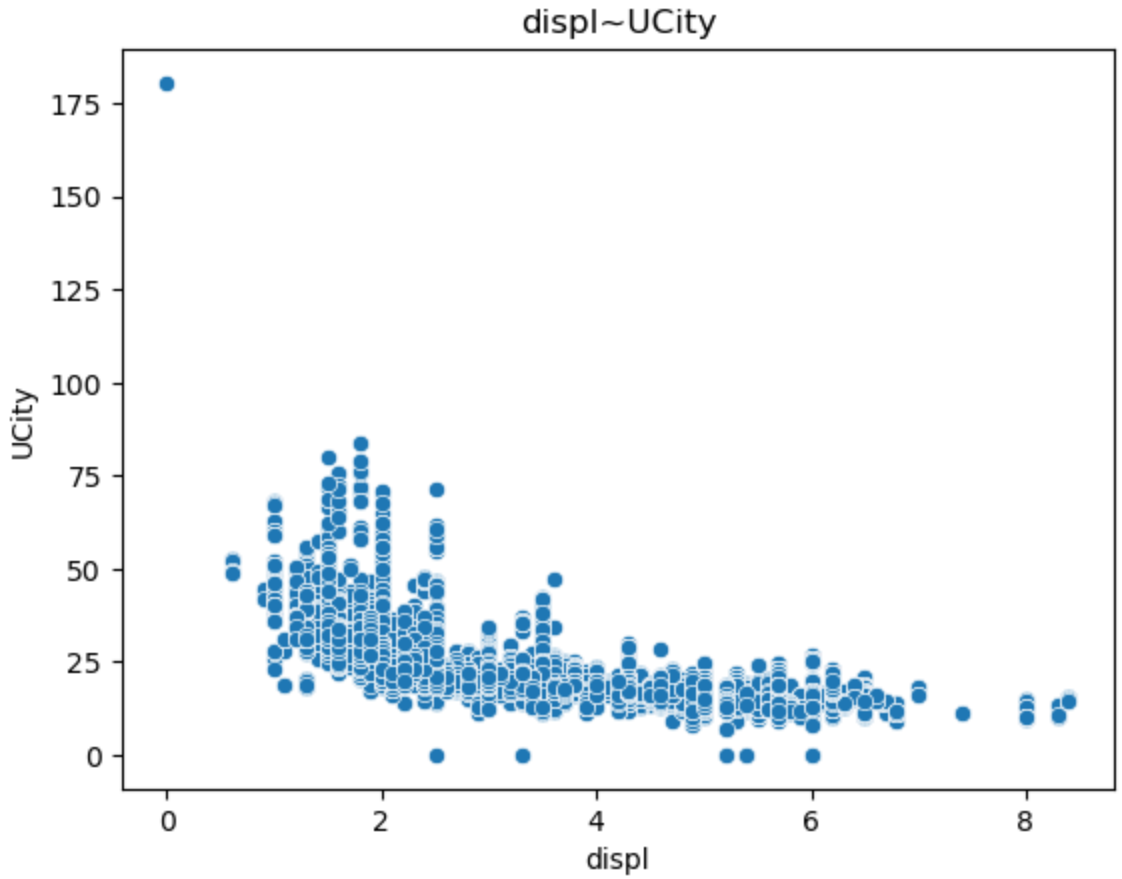
Out[600]:
```
0         2.0
1         4.9
2         2.2
3         5.2
4         2.2
         ...
40076     2.2
40077     2.2
40078     2.2
40079     2.2
40080     2.2
Name: displ, Length: 39912, dtype: float64
```

In [601…
```
# correlation between UCity and displ
print("Correlation between displ and UCity: ", df_displ.corr(df['UCity']))
```

```
Correlation between displ and UCity:  -0.7132493488120365
```

In [602…
```
# plot scatter graph to compare displ and UCity
sns.scatterplot(x = df_displ, y = df['UCity'])
plt.xlabel('displ')
plt.ylabel('UCity')
```

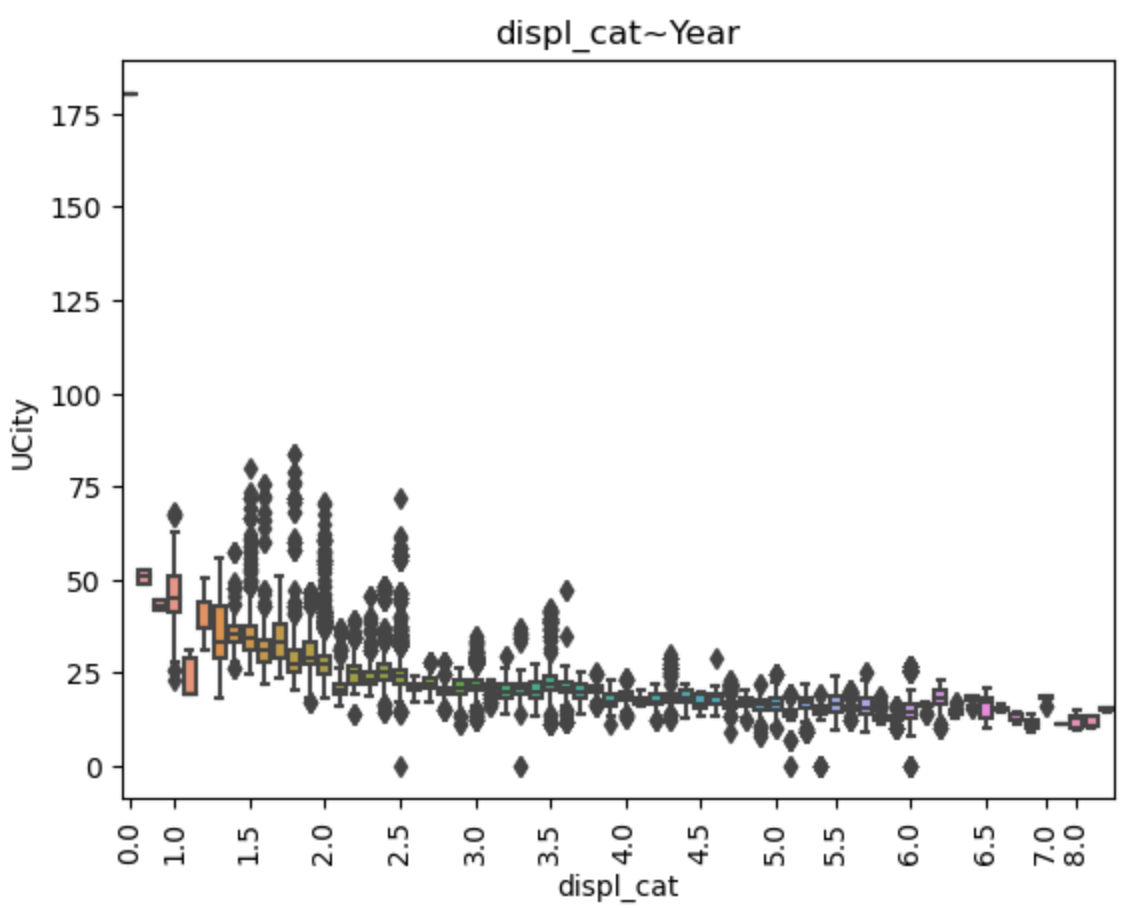
```
plt.title('displ~UCity')
plt.show()
```



In [603…

```
# box plot displ~UCity
ax = sns.boxplot(x = df_displ, y = df['UCity'])

# set title and redefine the xlabel
ax.set(title = "displ_cat~Year", xlabel = "displ_cat")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 0.5) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels, rotation = 90)
plt.show()
```

```
x_ticklabels: ['0.0', '0.6', '0.9', '1.0', '1.1', '1.2', '1.3', '1.4', '1.5', '1.6', '1.
7', '1.8', '1.9', '2.0', '2.1', '2.2', '2.3', '2.4', '2.5', '2.6', '2.7', '2.8', '2.9',
'3.0', '3.1', '3.2', '3.3', '3.4', '3.5', '3.6', '3.7', '3.8', '3.9', '4.0', '4.1', '4.
2', '4.3', '4.4', '4.5', '4.6', '4.7', '4.8', '4.9', '5.0', '5.2', '5.3', '5.4', '5.5',
'5.6', '5.7', '5.8', '5.9', '6.0', '6.1', '6.2', '6.3', '6.4', '6.5', '6.6', '6.7', '6.
8', '7.0', '7.4', '8.0', '8.3', '8.4']
```



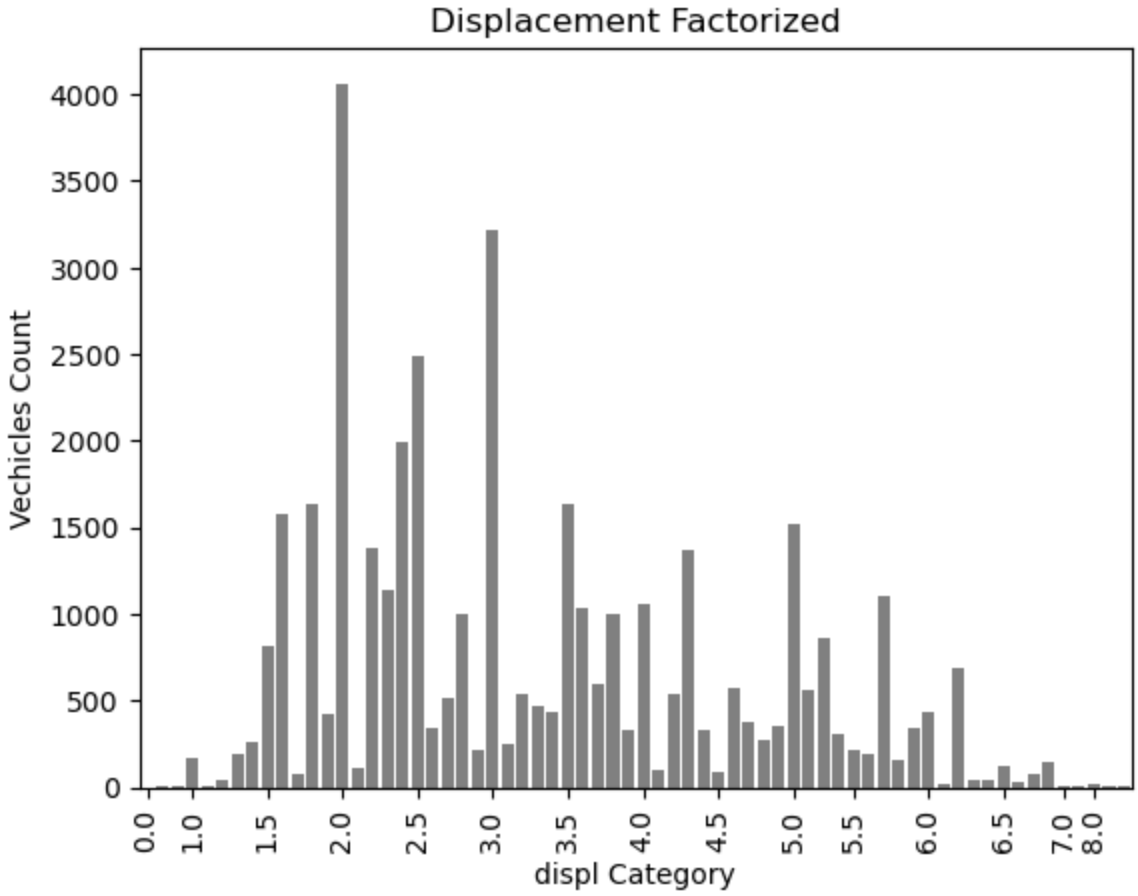
In [604…

```
# plot count graph for displ~vechicles

ax = sns.countplot(x = df_displ, color = 'grey')
ax.set(title = "Displacement Factorized", xlabel = "displ Category",
       ylabel = "Vechicles Count")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 0.5) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels, rotation = 90)
```

Out[604]:

```
([<matplotlib.axis.XTick at 0x1f17e0fe6d0>,
  <matplotlib.axis.XTick at 0x1f17e07ce50>,
  <matplotlib.axis.XTick at 0x1f179b55890>,
  <matplotlib.axis.XTick at 0x1f17e1ca7d0>,
  <matplotlib.axis.XTick at 0x1f17e1d0490>,
  <matplotlib.axis.XTick at 0x1f17e1d2b90>,
  <matplotlib.axis.XTick at 0x1f17e1d3d10>,
  <matplotlib.axis.XTick at 0x1f17e1d5f10>,
  <matplotlib.axis.XTick at 0x1f17e1d81d0>,
  <matplotlib.axis.XTick at 0x1f17e1da350>,
  <matplotlib.axis.XTick at 0x1f17e1e0490>,
  <matplotlib.axis.XTick at 0x1f17dfbc450>,
  <matplotlib.axis.XTick at 0x1f17e1e2f50>,
  <matplotlib.axis.XTick at 0x1f17e1f0fd0>,
  <matplotlib.axis.XTick at 0x1f17e1f3250>],
```

```
 [Text(0, 0, '0.0'),
  Text(3, 0, '1.0'),
  Text(8, 0, '1.5'),
  Text(13, 0, '2.0'),
  Text(18, 0, '2.5'),
  Text(23, 0, '3.0'),
  Text(28, 0, '3.5'),
  Text(33, 0, '4.0'),
  Text(38, 0, '4.5'),
  Text(43, 0, '5.0'),
  Text(47, 0, '5.5'),
  Text(52, 0, '6.0'),
  Text(57, 0, '6.5'),
  Text(61, 0, '7.0'),
  Text(63, 0, '8.0')])
```

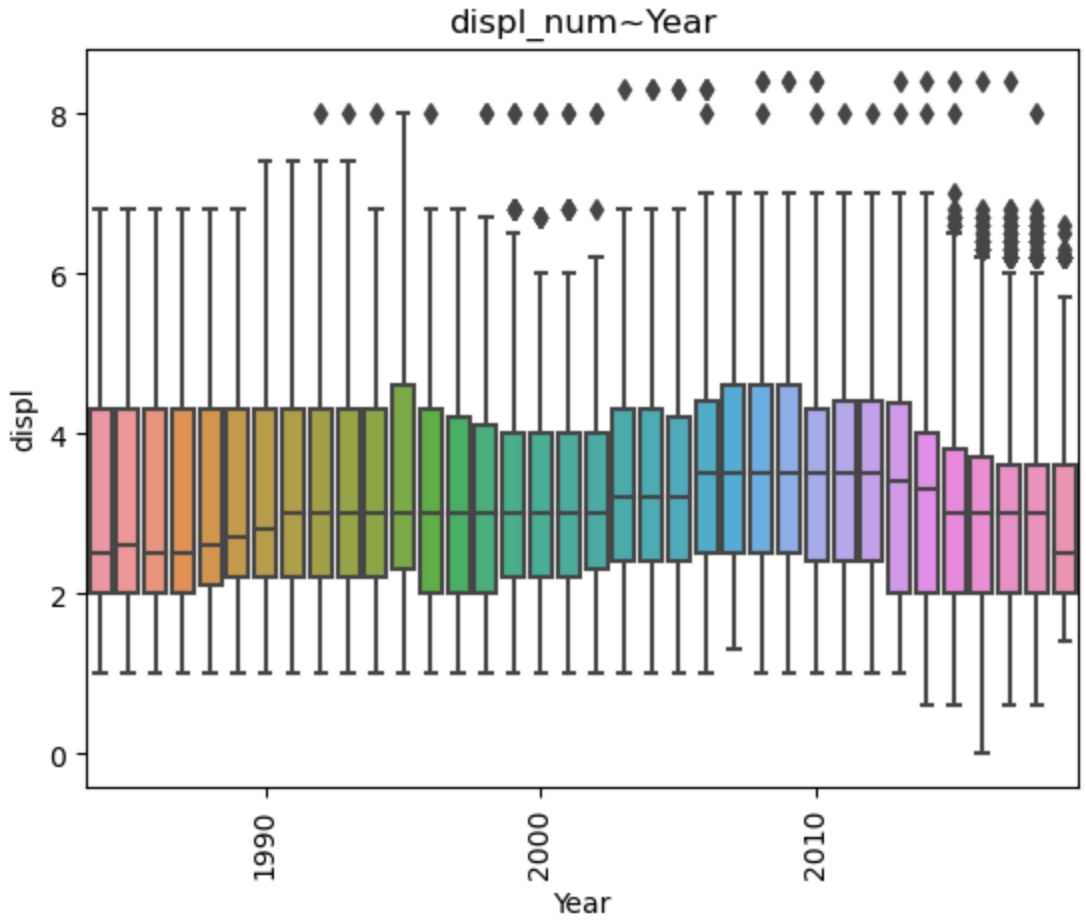


In [605…

```python
# box plot displ~year
ax = sns.boxplot(x = df['year'], y = df_displ)

# set title and redefine the xlabel
ax.set(title = "displ_num~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])
```

```
# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels, rotation = 90)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



Numer - cityXXXX

In [606…
```
# correlation between UCity and city08
print("Correlation between city08 and UCity: ", df['city08'].corr(df['UCity']))
```

```
Correlation between city08 and UCity:  0.9971665594566513
```

In [607…
```
# correlation between UCity and city08U
print("Correlation between city08U and UCity: ", df['city08U'].corr(df['UCity']))
```

```
Correlation between city08U and UCity:  0.6384905422377316
```

In [608…
```
# correlation between UCity and cityA08
print("Correlation between cityA08 and UCity: ", df['cityA08'].corr(df['UCity']))
```

```
Correlation between cityA08 and UCity:  0.07445808413014819
```

In [609…
```
# correlation between UCity and cityA08U
print("Correlation between cityA08U and UCity: ", df['cityA08U'].corr(df['UCity']))
```

```
Correlation between cityA08U and UCity:  0.09093533684198754
```

In [610…
```
# correlation between UCity and cityE
print("Correlation between cityE and UCity: ", df['cityE'].corr(df['UCity']))
```

```
Correlation between cityE and UCity:  0.49012478827405825
```

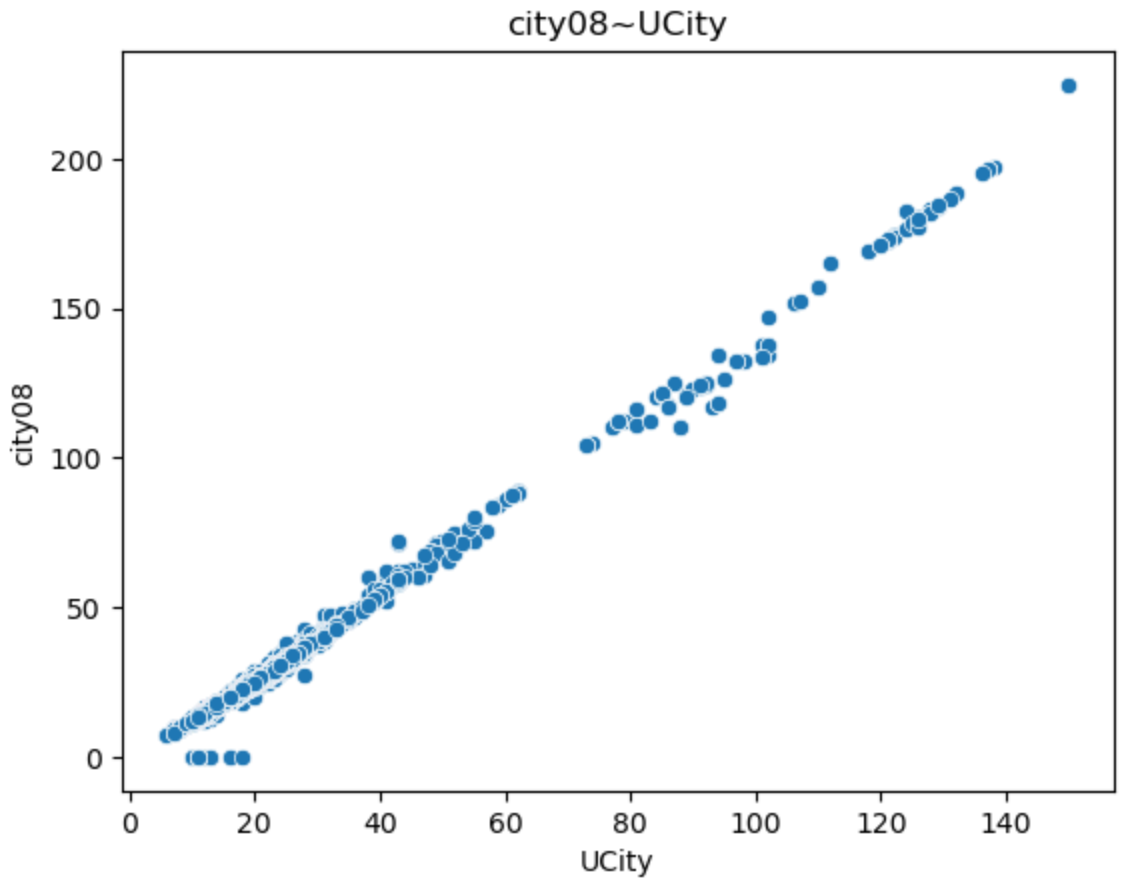
In [611…
```
# correlation between UCity and cityUF
print("Correlation between cityUF and UCity: ", df['cityUF'].corr(df['UCity']))
```

```
Correlation between cityUF and UCity:  0.09891092041539981
```

In [612…
```
# correlation between UCity and phevCity
print("Correlation between phevCity and UCity: ", df['phevCity'].corr(df['UCity']))
```

```
Correlation between phevCity and UCity:  0.10981464263302597
```

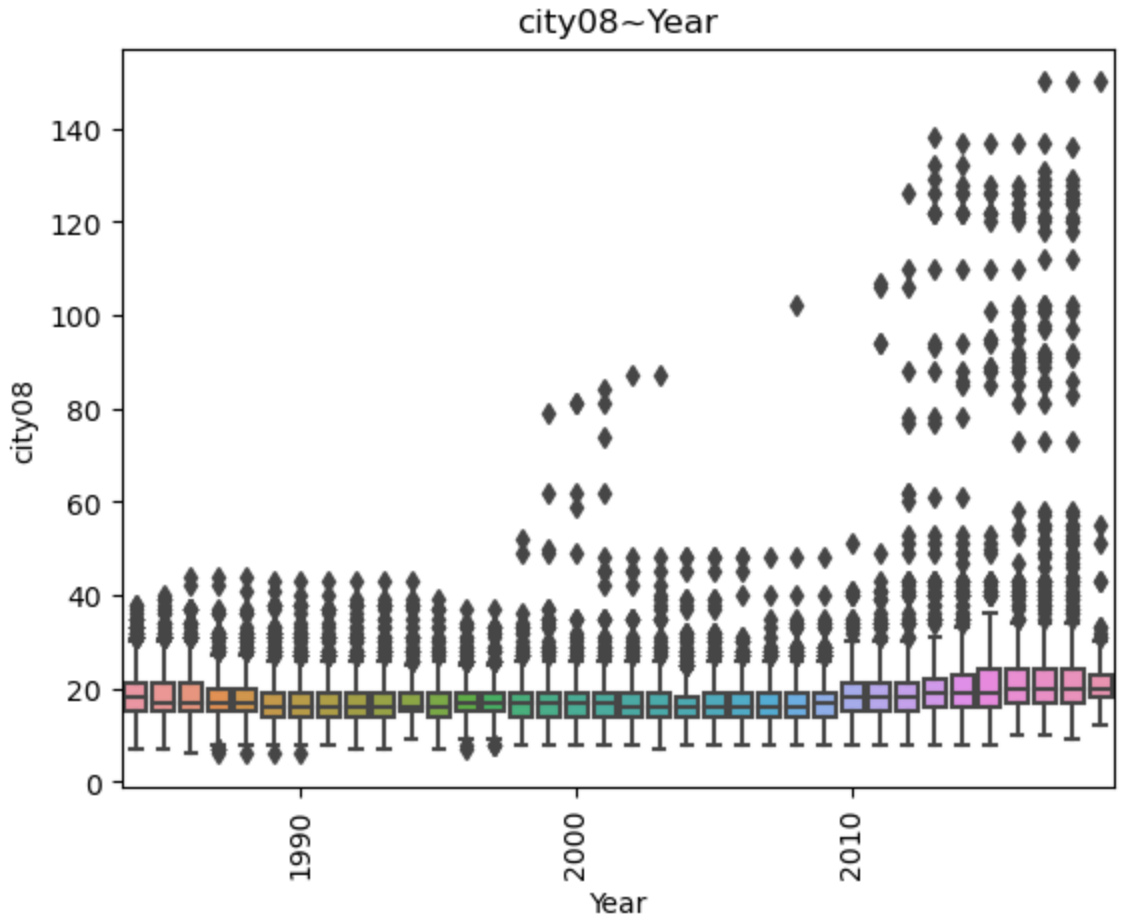
In [613…
```
# plot scatter graph to compare city08 and UCity
sns.scatterplot(x = df['city08'], y = df['UCity'])
plt.xlabel('UCity')
plt.ylabel('city08')
plt.title('city08~UCity')
plt.show()
```



In [614…
```
# box plot city08~year
ax = sns.boxplot(x = df['year'], y = df['city08'])

# set title and redefine the xlabel
ax.set(title = "city08~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])
```

```
# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels, rotation = 90)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```

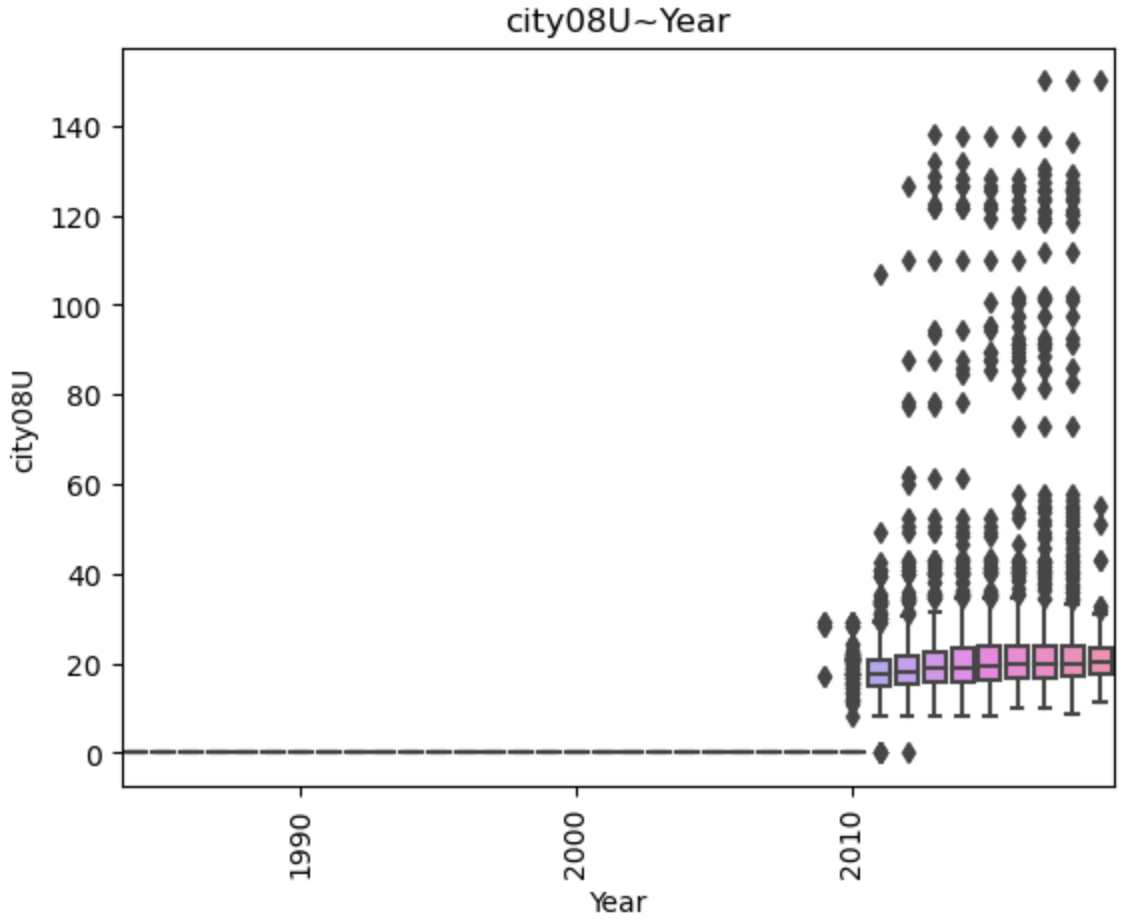


In [615…

```
# box plot city08U~year
ax = sns.boxplot(x = df['year'], y = df['city08U'])

# set title and redefine the xlabel
ax.set(title = "city08U~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels, rotation = 90)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
```

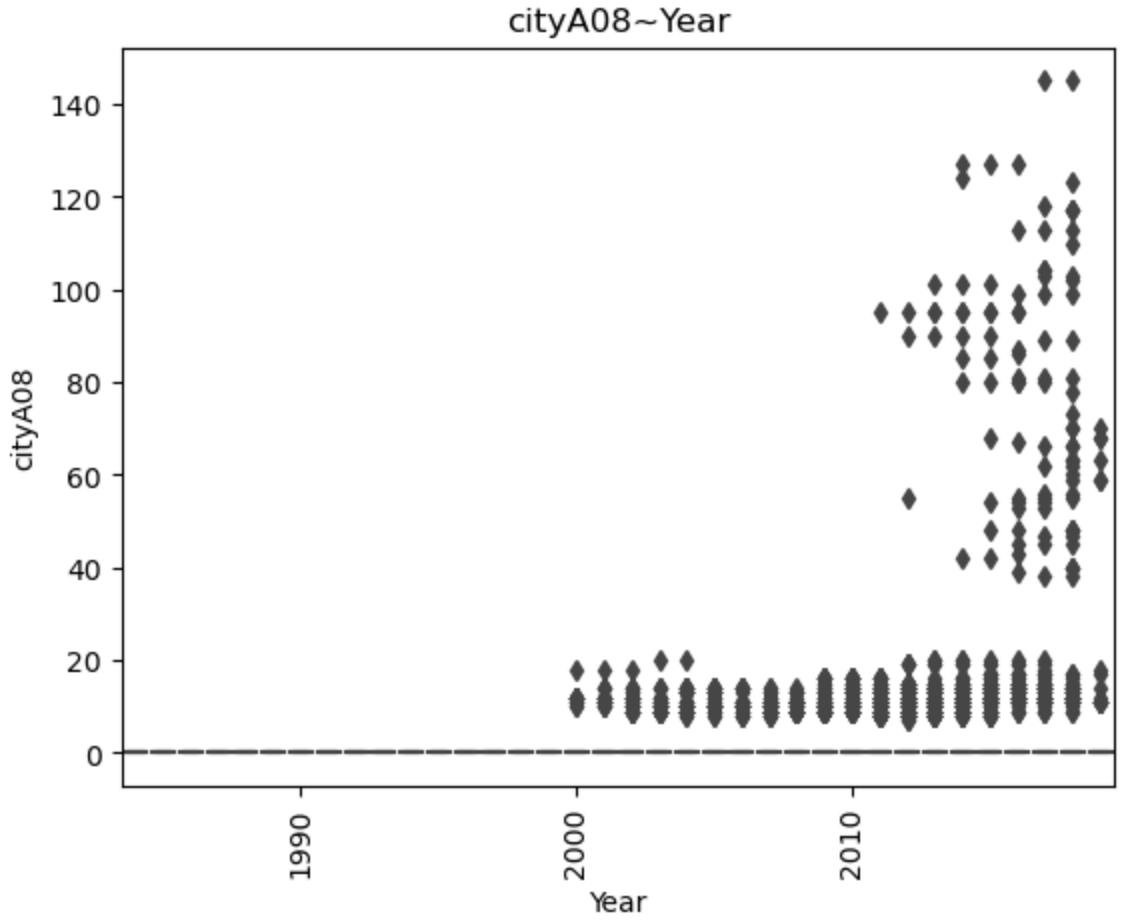
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']



In [616…

```
# box plot cityA08~year
ax = sns.boxplot(x = df['year'], y = df['cityA08'])

# set title and redefine the xlabel
ax.set(title = "cityA08~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels, rotation = 90)
plt.show()
```

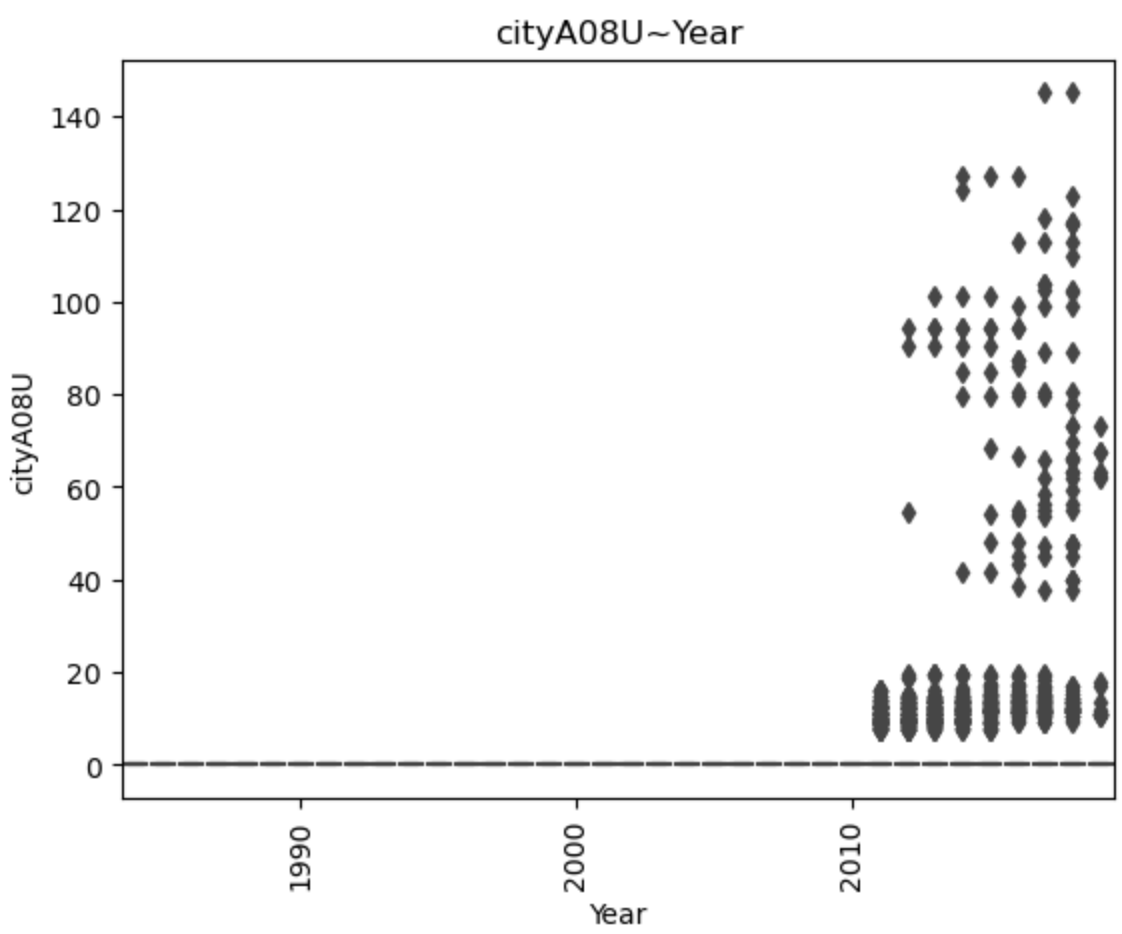
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



In [617…

```
# box plot cityA08U~year
ax = sns.boxplot(x = df['year'], y = df['cityA08U'])

# set title and redefine the xlabel
ax.set(title = "cityA08U~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels, rotation = 90)
plt.show()
```

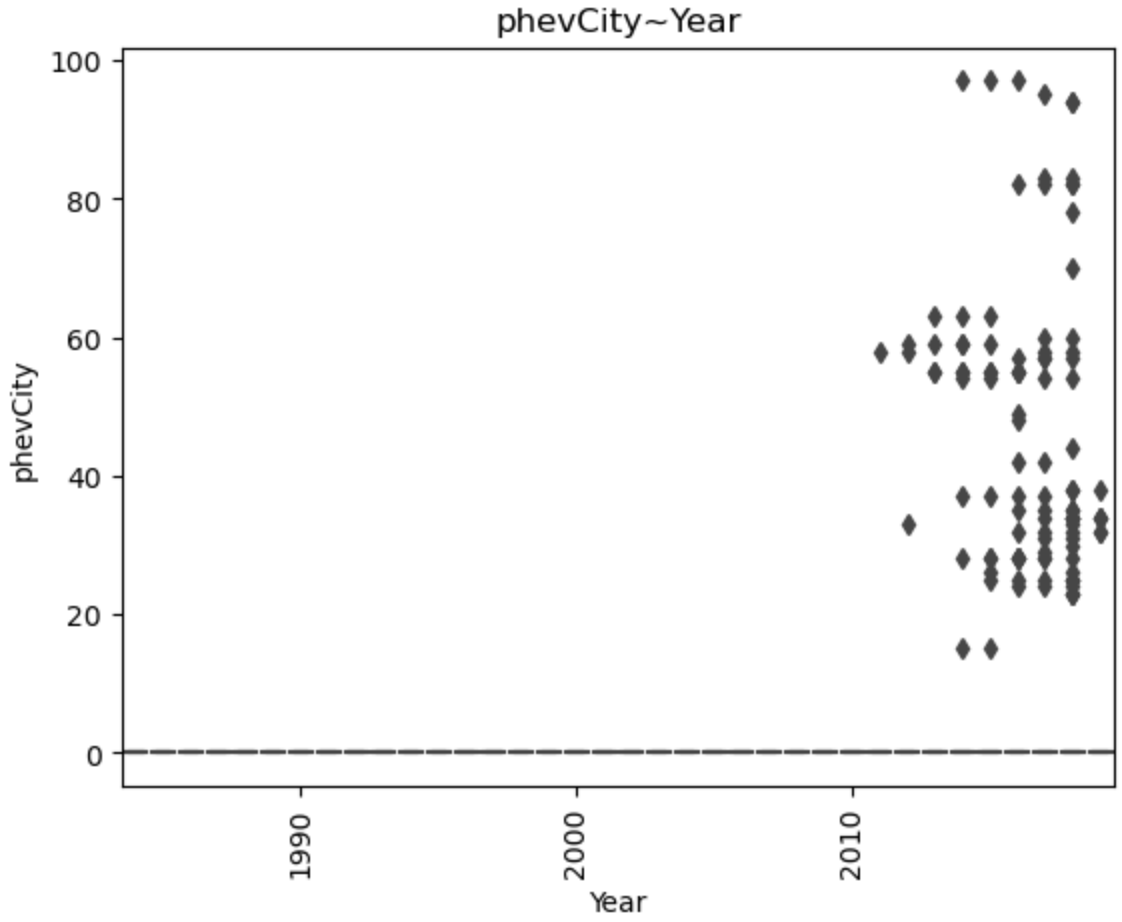
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



In [618…

```
# box plot phevCity~year
ax = sns.boxplot(x = df['year'], y = df['phevCity'])

# set title and redefine the xlabel
ax.set(title = "phevCity~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels, rotation = 90)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



Numeric - feScore

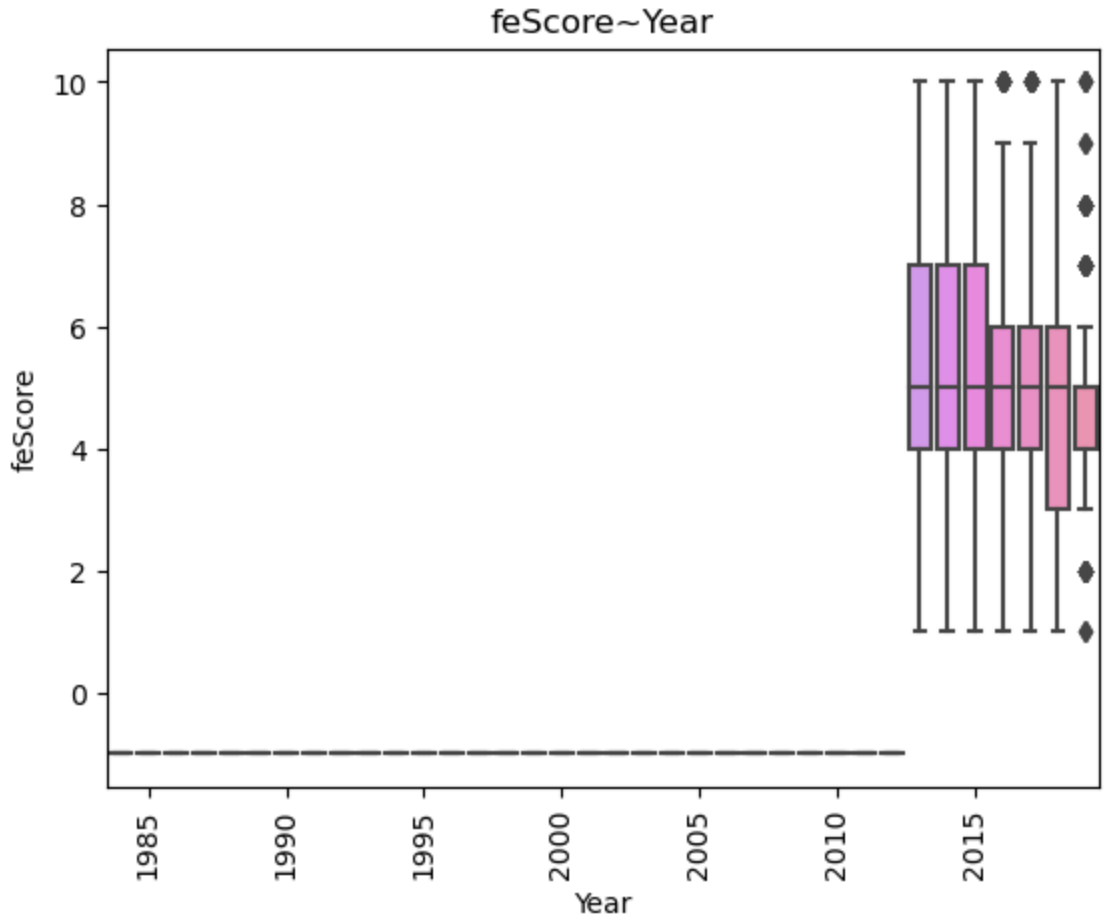
In [619…
```
df['feScore'].describe()
```

Out[619]:
```
count    40081.000000
mean         0.238891
std          2.609123
min         -1.000000
25%         -1.000000
50%         -1.000000
75%         -1.000000
max         10.000000
Name: feScore, dtype: float64
```

In [620…
```
# box plot phevCity~year
ax = sns.boxplot(x = df['year'], y = df['feScore'])

# set title and redefine the xlabel
ax.set(title = "feScore~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 5) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])
```

```
# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels, rotation = 90)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



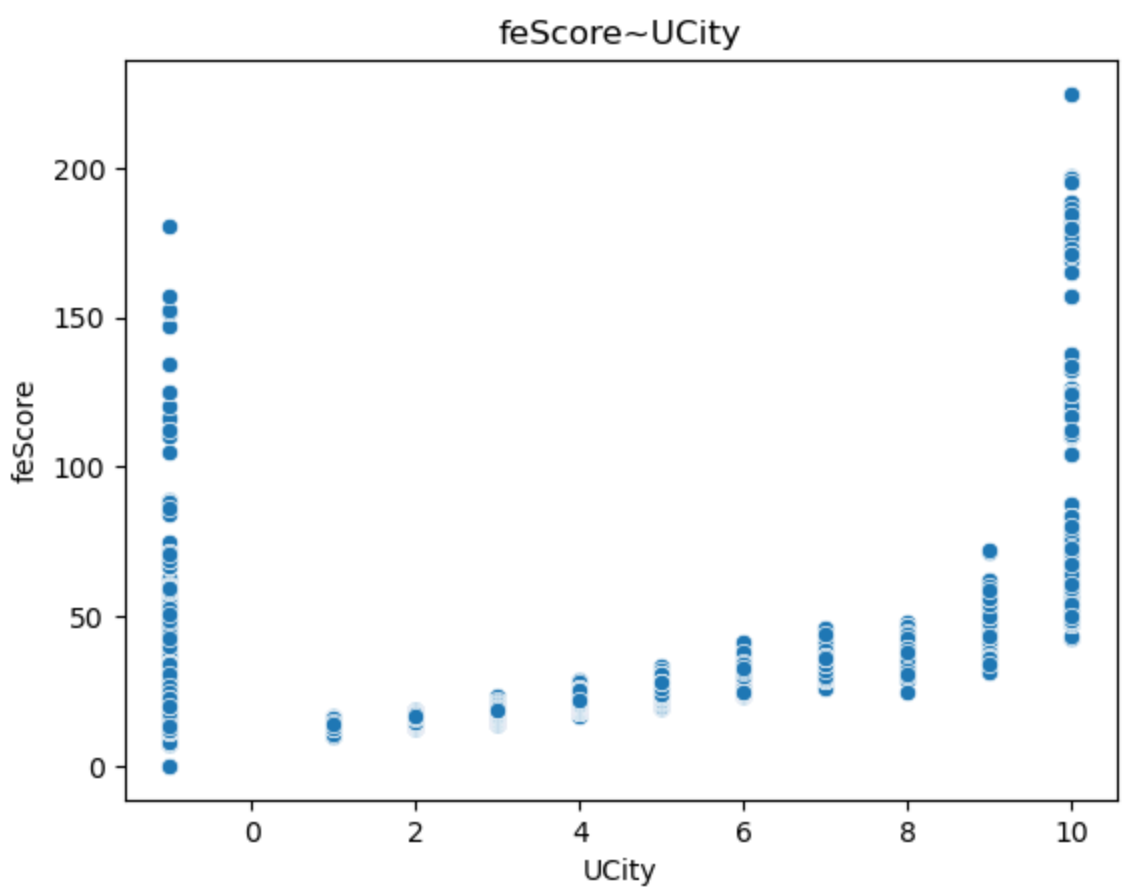
In [621…

```
# correlation between UCity and feScore
print("Correlation between feScore and UCity: ", df['feScore'].corr(df['UCity']))
```

```
Correlation between feScore and UCity:  0.3978344569411843
```

In [622…

```
# plot scatter graph to compare feScore and UCity
sns.scatterplot(x = df['feScore'], y = df['UCity'])
plt.xlabel('UCity')
plt.ylabel('feScore')
plt.title('feScore~UCity')
plt.show()
```



Numeric - ghgScoreX

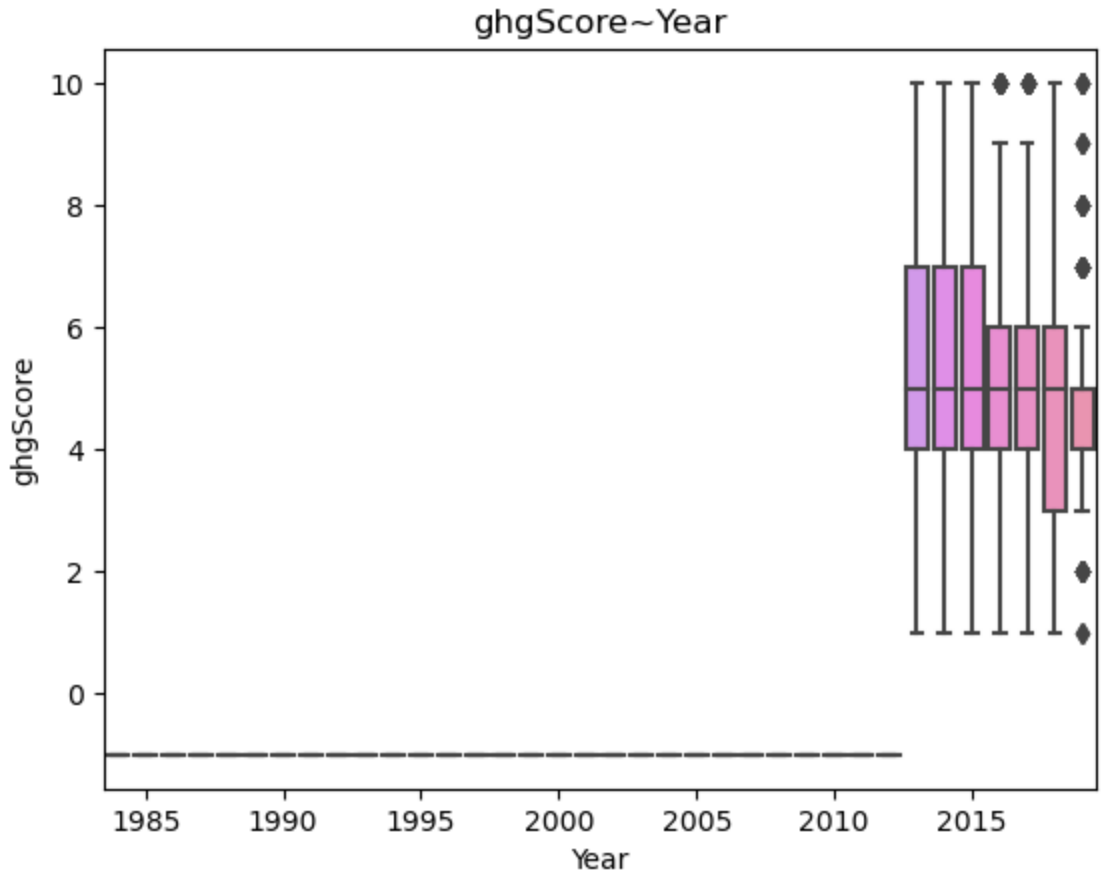
In [623…
```
df['ghgScore'].describe()
```

Out[623]:
```
count    40081.000000
mean         0.237020
std          2.605921
min         -1.000000
25%         -1.000000
50%         -1.000000
75%         -1.000000
max         10.000000
Name: ghgScore, dtype: float64
```

In [624…
```
# box plot ghgScore~year
ax = sns.boxplot(x = df['year'], y = df['ghgScore'])

# set title and redefine the xlabel
ax.set(title = "ghgScore~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 5) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
```

```
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```

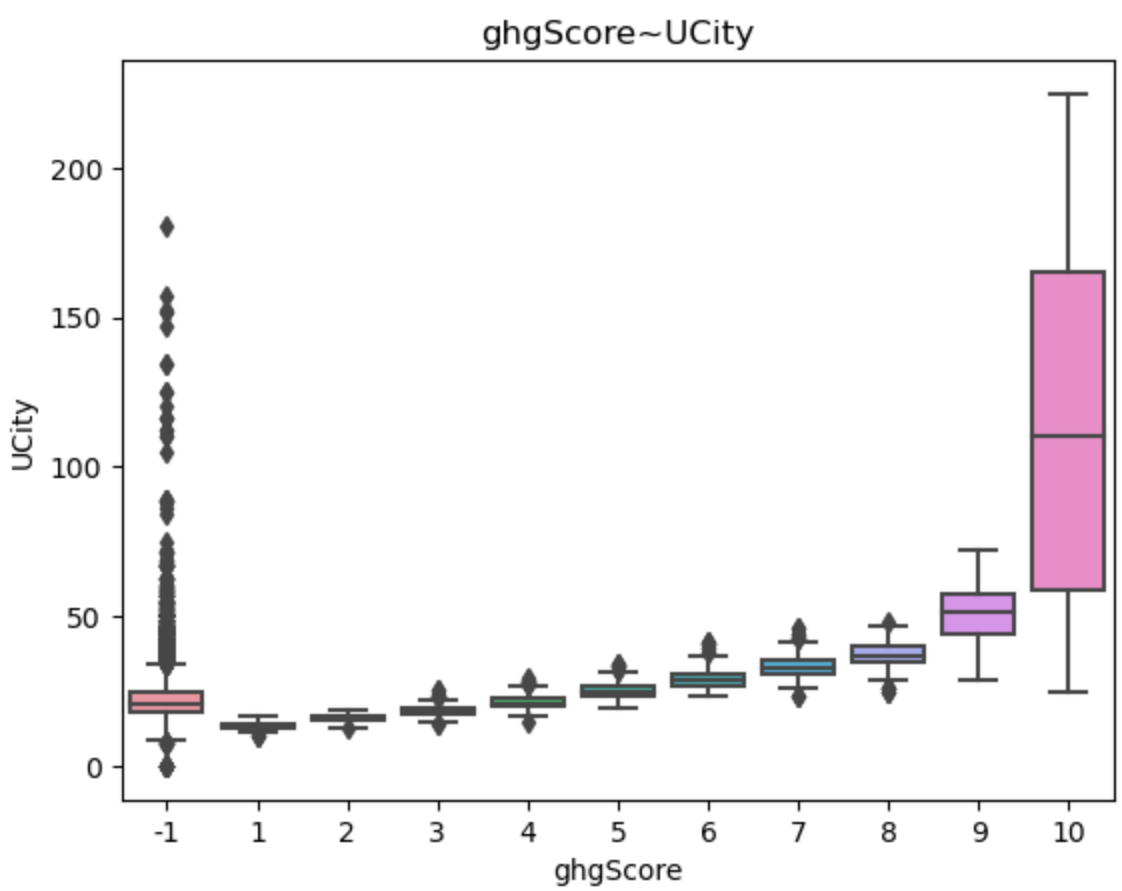


In [625…
```
# correlation between UCity and ghgScore
print("Correlation between ghgScore and UCity: ", df['ghgScore'].corr(df['UCity']))
```

```
Correlation between ghgScore and UCity:  0.397377591709909
```

In [626…
```
# box plot ghgScore~UCity
ax = sns.boxplot(x = df['ghgScore'], y = df['UCity'])

# set title and redefine the xlabel
ax.set(title = "ghgScore~UCity", xlabel = "ghgScore")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 1) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['-1', '1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
```



In [627…
```
# correlation between feScore and ghgScore
print("Correlation between ghgScore and feScore: ", df['ghgScore'].corr(df['feScore']))
```

```
Correlation between ghgScore and feScore:  0.9993790177391131
```

Numeric - barrels08 and barrelsA08

In [628…
```
df['barrels08'].describe()
```

Out[628]:
```
count    40081.000000
mean        17.363564
std          4.597119
min          0.060000
25%         14.330870
50%         16.480500
75%         19.388824
max         47.087143
Name: barrels08, dtype: float64
```

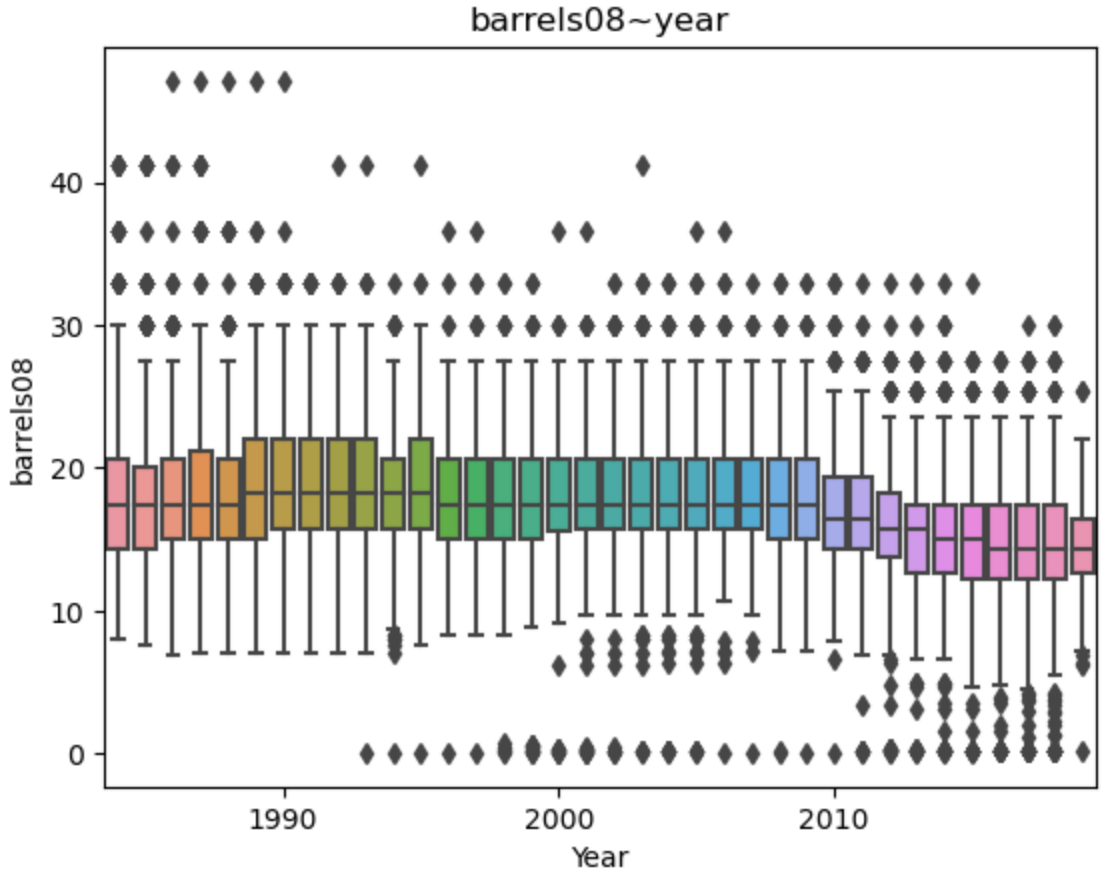
In [629…
```
df['barrelsA08'].describe()
```

Out[629]:
```
count    40081.000000
mean         0.220069
std          1.143270
min          0.000000
25%          0.000000
50%          0.000000
75%          0.000000
max         18.311667
Name: barrelsA08, dtype: float64
```

In [630…
```
# box plot barrels08~year
ax = sns.boxplot(x = df['year'], y = df['barrels08'])

# set title and redefine the xlabel
ax.set(title = "barrels08~year", xlabel = "Year")
```

```
# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```

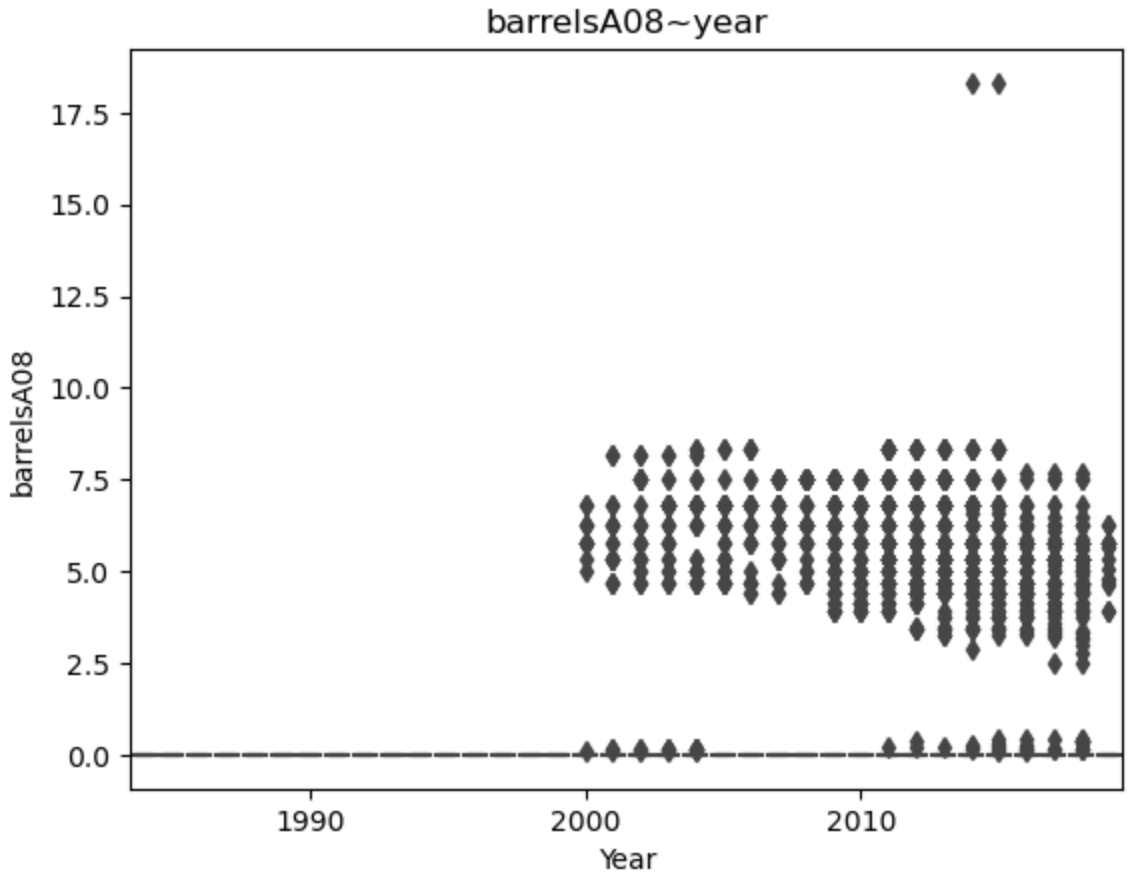


In [631…

```
# box plot barrelsA08~year
ax = sns.boxplot(x = df['year'], y = df['barrelsA08'])

# set title and redefine the xlabel
ax.set(title = "barrelsA08~year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
```

```
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



Numeric - evMotor

In [632…
```
df['evMotor'].describe()
```

Out[632]:
```
count             40081
unique              141
top       Not Available
freq              39345
Name: evMotor, dtype: object
```

In [633…
```
df['evMotor'].value_counts()
```

Out[633]:
```
Not Available                                          39345
288V Ni-MH                                               122
245V Ni-MH                                                48
144V Li-Ion                                               28
330V Ni-MH                                                26
                                                       ...
180 and 350 kW AC Induction (85 kW-hr battery pack)        1
Two 480V Li-Ion                                            1
132 kW AC Synchronous                                      1
```

```
132kW                                                   1
147 and 211 kW AC 3-Phase                               1
Name: evMotor, Length: 141, dtype: int64
```

Numeric - combXXXX

In [634…
```
# correlation between comb08 and UCity
print("Correlation between comb08 and UCity: ", df['comb08'].corr(df['UCity']))
```

```
Correlation between comb08 and UCity:  0.9839859069971367
```

In [635…
```
# correlation between comb08U and UCity
print("Correlation between comb08U and UCity: ", df['comb08U'].corr(df['UCity']))
```

```
Correlation between comb08U and UCity:  0.587552436185925
```

In [636…
```
# correlation between combA08 and UCity
print("Correlation between combA08 and UCity: ", df['combA08'].corr(df['UCity']))
```

```
Correlation between combA08 and UCity:  0.06699893400661679
```

In [637…
```
# correlation between combA08U and UCity
print("Correlation between combA08U and UCity: ", df['combA08U'].corr(df['UCity']))
```

```
Correlation between combA08U and UCity:  0.08585675366543616
```

In [638…
```
# correlation between combE and UCity
print("Correlation between combE and UCity: ", df['combE'].corr(df['UCity']))
```

```
Correlation between combE and UCity:  0.5048838736069216
```

In [639…
```
# correlation between combinedCD and UCity
print("Correlation between combinedCD and UCity: ", df['combinedCD'].corr(df['UCity']))
```

```
Correlation between combinedCD and UCity:  0.0025209779342005886
```

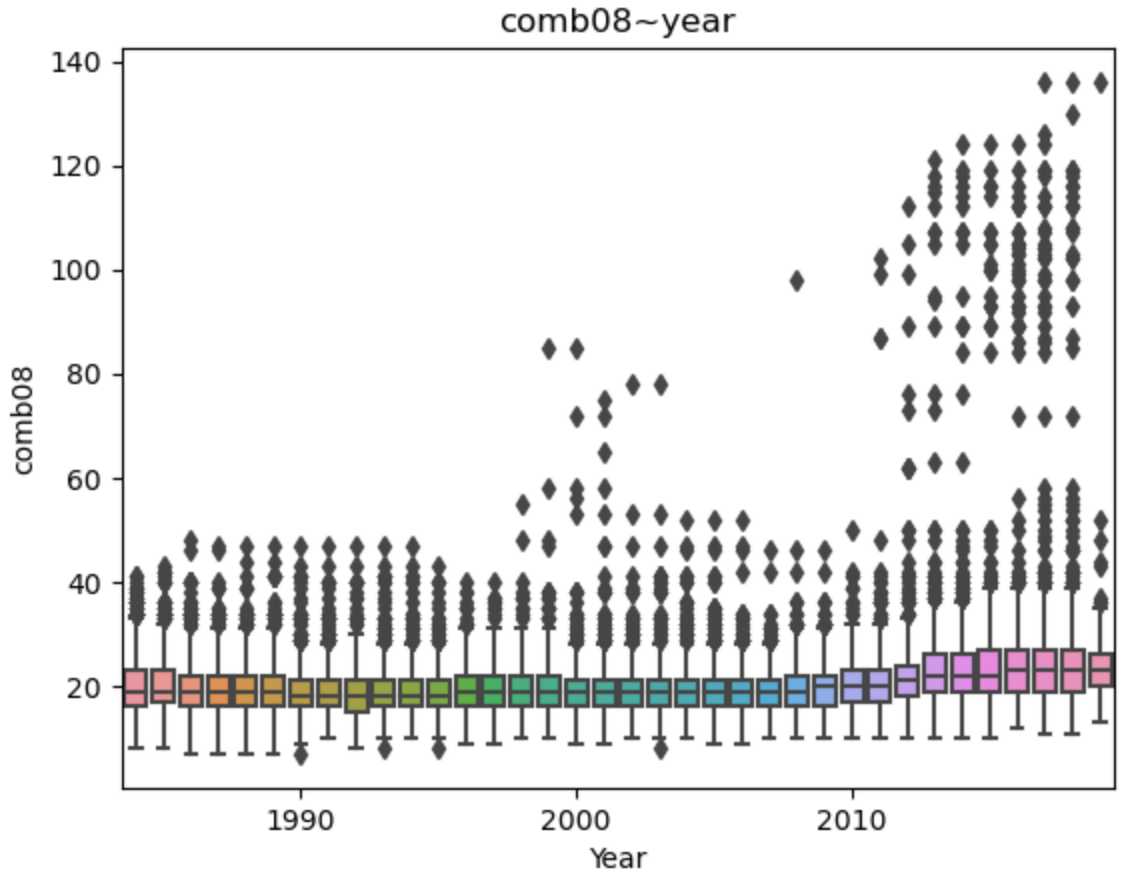
In [640…
```
# correlation between combinedUF and UCity
print("Correlation between combinedUF and UCity: ", df['combinedUF'].corr(df['UCity']))
```

```
Correlation between combinedUF and UCity:  0.0979101234087332
```

In [641…
```
# box plot comb08~year
ax = sns.boxplot(x = df['year'], y = df['comb08'])

# set title and redefine the xlabel
ax.set(title = "comb08~year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
```

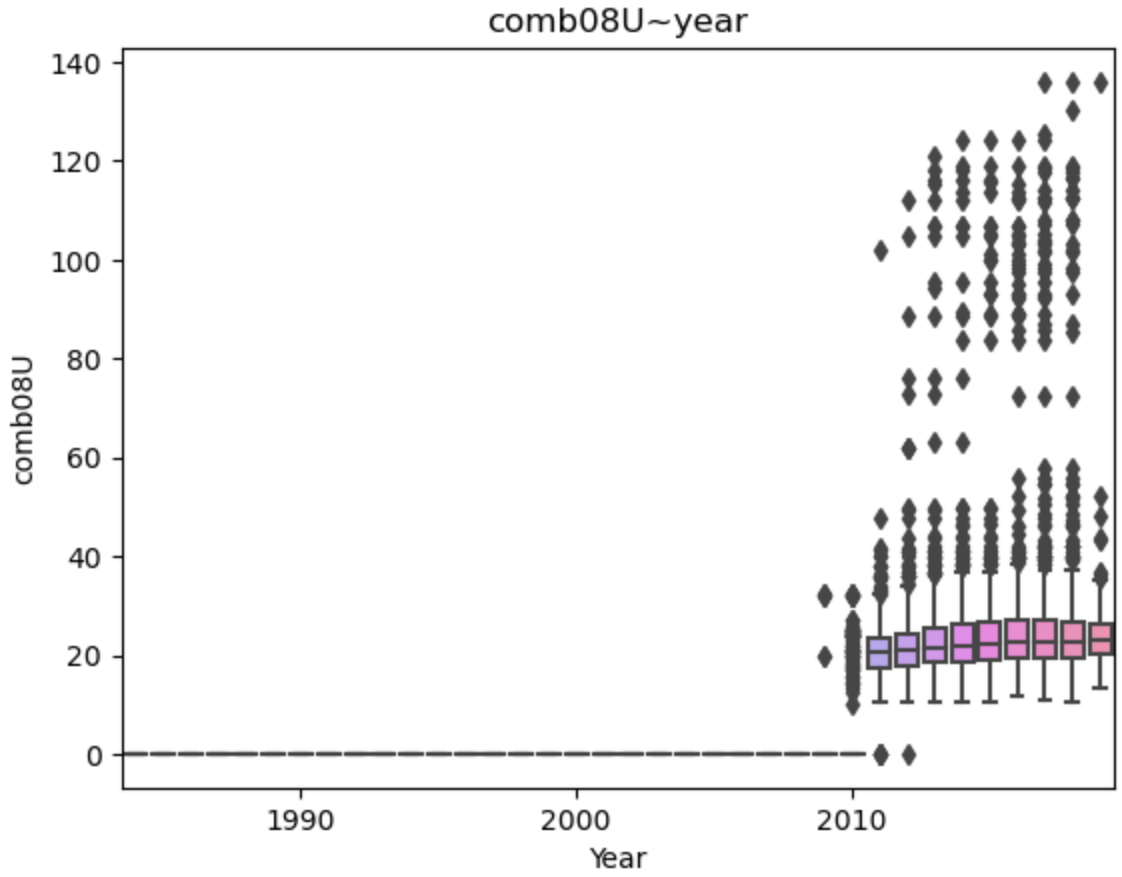
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']



In [642…

```
# box plot comb08U~year
ax = sns.boxplot(x = df['year'], y = df['comb08U'])

# set title and redefine the xlabel
ax.set(title = "comb08U~year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

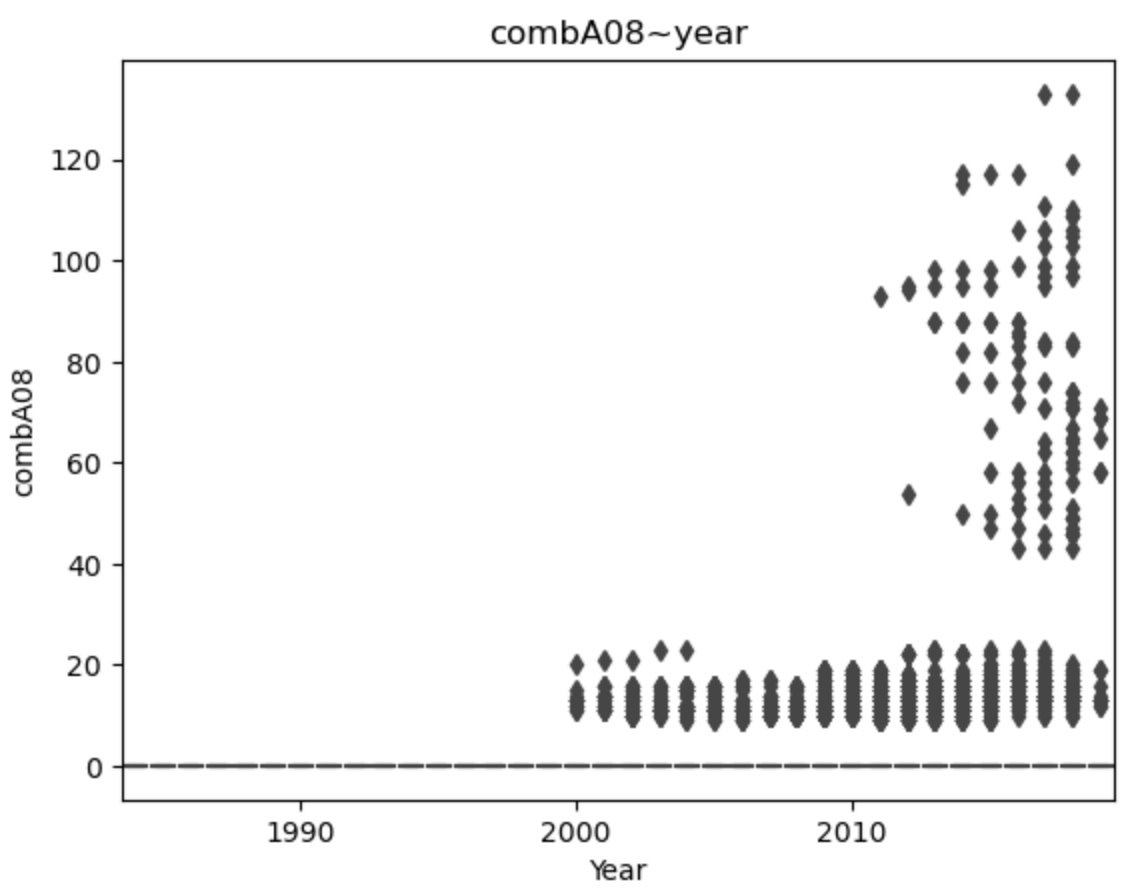
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']



In [643…

```
# box plot combA08~year
ax = sns.boxplot(x = df['year'], y = df['combA08'])

# set title and redefine the xlabel
ax.set(title = "combA08~year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

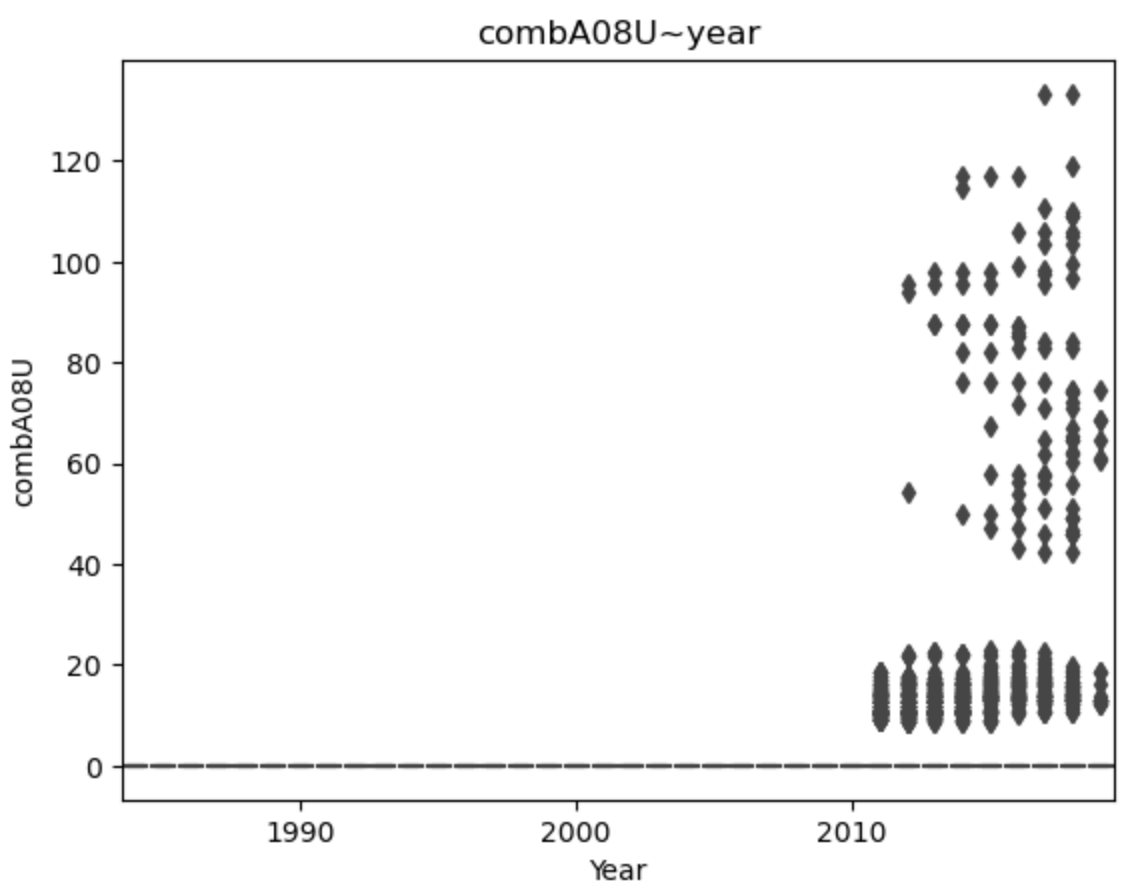
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



In [644…

```
# box plot combA08U~year
ax = sns.boxplot(x = df['year'], y = df['combA08U'])

# set title and redefine the xlabel
ax.set(title = "combA08U~year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

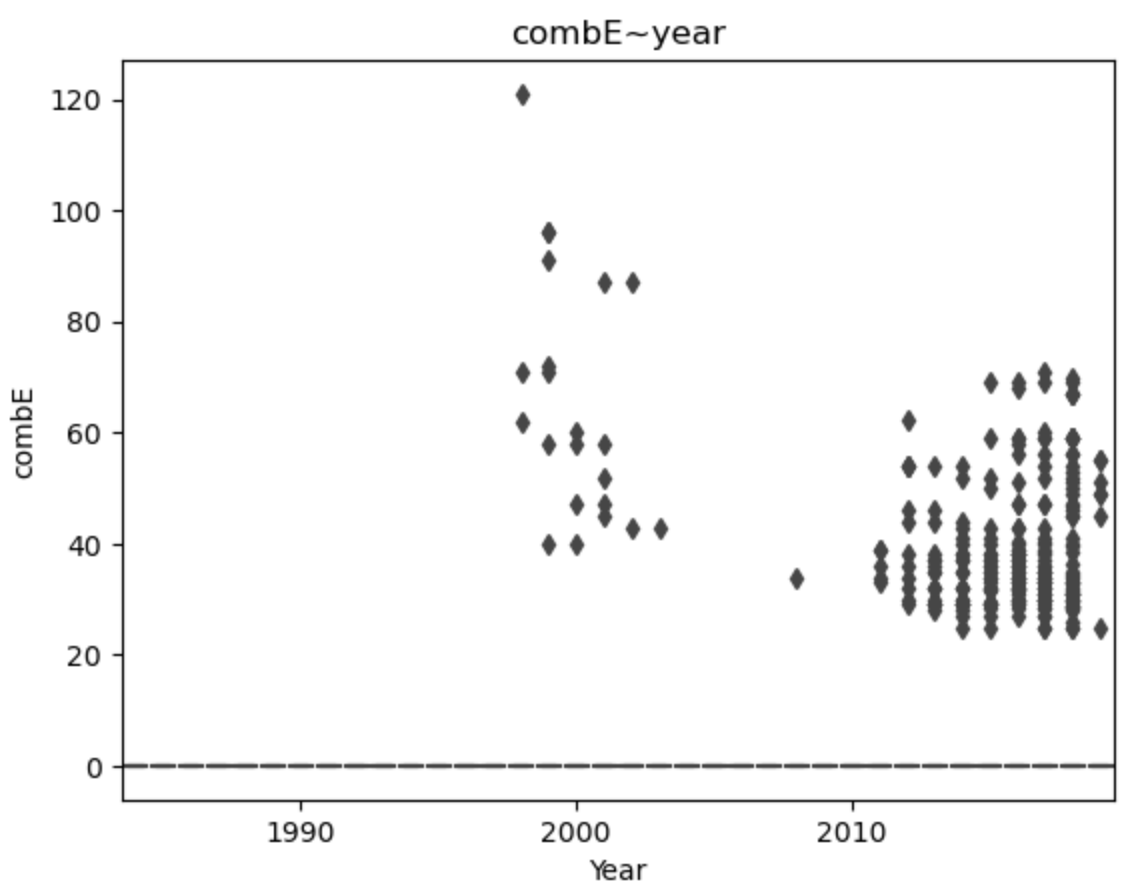
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



In [645…

```
# box plot combE~year
ax = sns.boxplot(x = df['year'], y = df['combE'])

# set title and redefine the xlabel
ax.set(title = "combE~year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

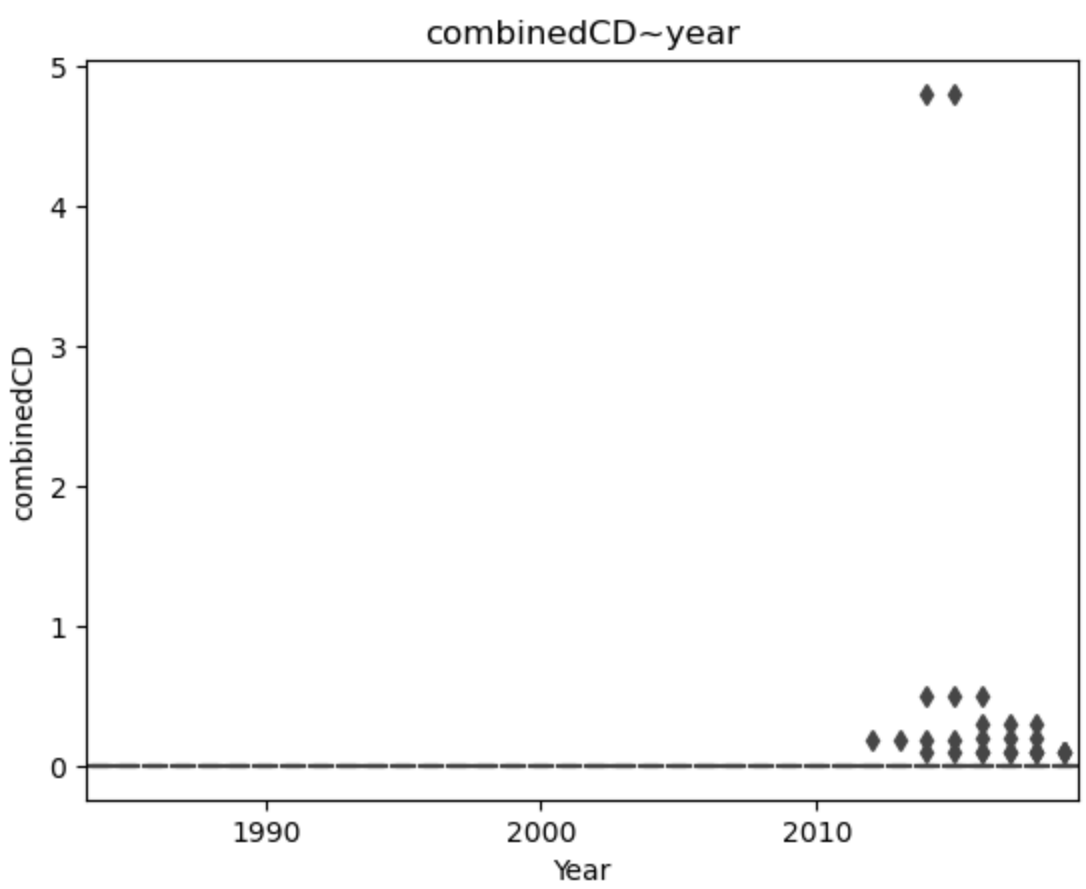
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



In [646…

```
# box plot combinedCD~year
ax = sns.boxplot(x = df['year'], y = df['combinedCD'])

# set title and redefine the xlabel
ax.set(title = "combinedCD~year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

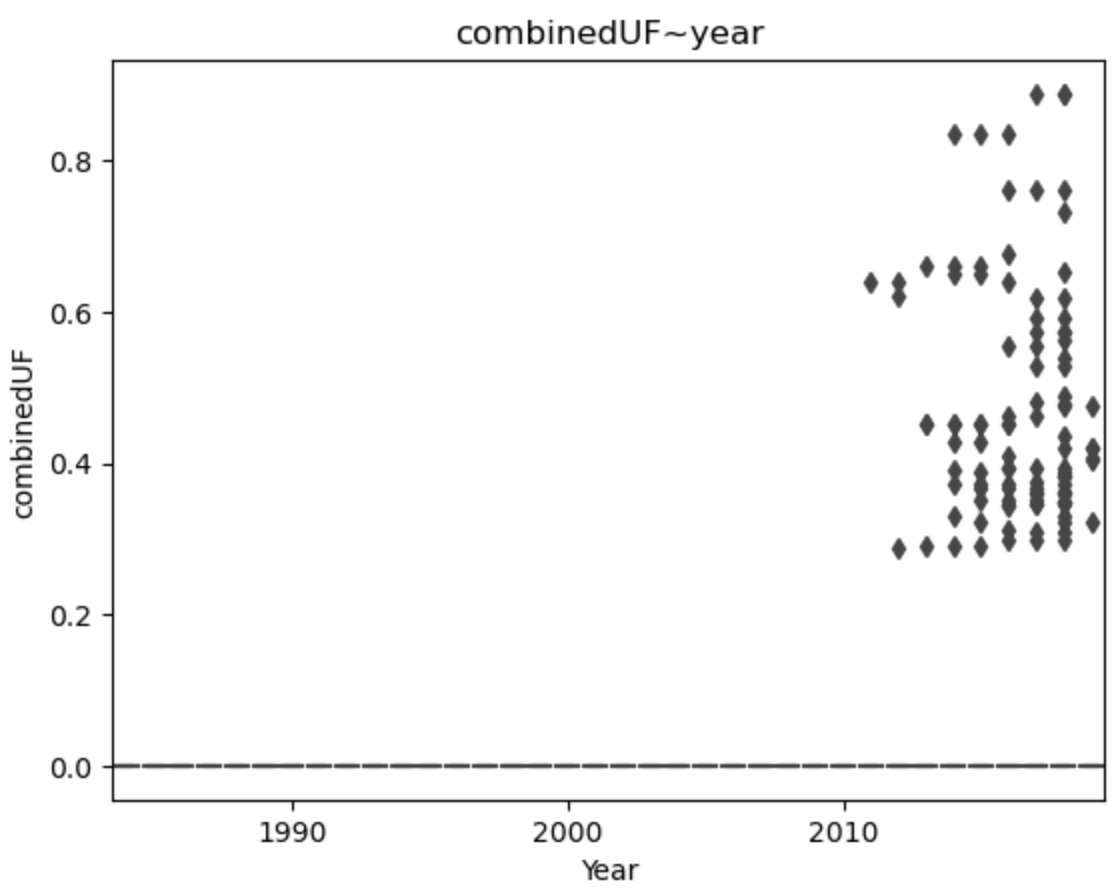
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



In [647…

```
# box plot combinedUF~year
ax = sns.boxplot(x = df['year'], y = df['combinedUF'])

# set title and redefine the xlabel
ax.set(title = "combinedUF~year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (float(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```



Categorical + Logical - drive

In [648…
```
df['drive'].describe()
```

Out[648]:
```
count                  40081
unique                     9
top       Front-Wheel Drive
freq                   13939
Name: drive, dtype: object
```

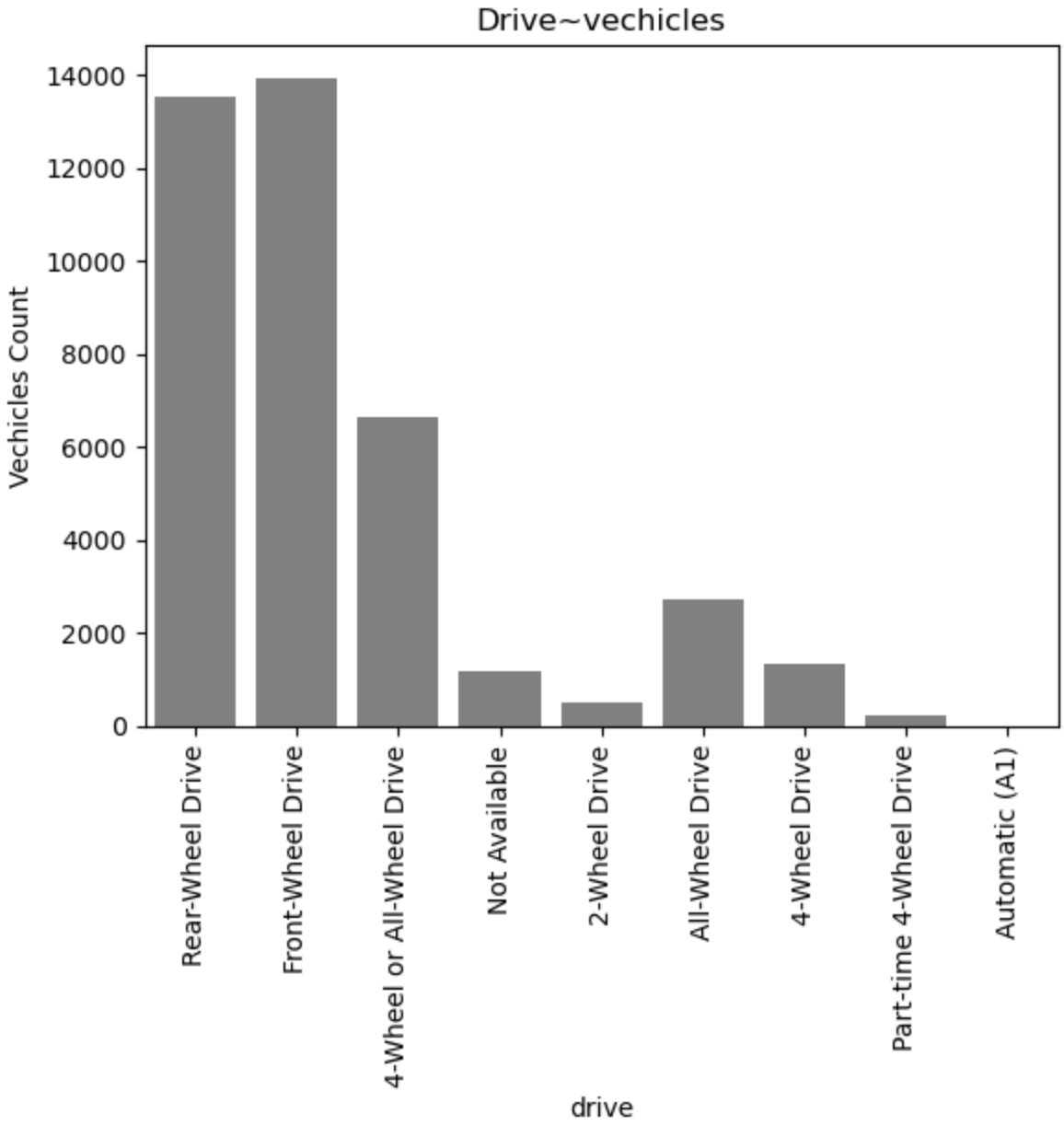
In [649…
```
df['drive'].value_counts()
```

Out[649]:
```
Front-Wheel Drive             13939
Rear-Wheel Drive              13539
4-Wheel or All-Wheel Drive     6648
All-Wheel Drive                2713
4-Wheel Drive                  1328
Not Available                  1189
2-Wheel Drive                   507
Part-time 4-Wheel Drive         217
Automatic (A1)                    1
Name: drive, dtype: int64
```

In [650…
```
# plot count graph for drive~vechicles

ax = sns.countplot(x = df['drive'], color = 'grey')
ax.set(title = "Drive~vechicles", xlabel = "drive",
       ylabel = "Vechicles Count")

plt.xticks(rotation = 90)
```

Out[650]:
```
(array([0, 1, 2, 3, 4, 5, 6, 7, 8]),
 [Text(0, 0, 'Rear-Wheel Drive'),
  Text(1, 0, 'Front-Wheel Drive'),
  Text(2, 0, '4-Wheel or All-Wheel Drive'),
  Text(3, 0, 'Not Available'),
```

```
  Text(4, 0, '2-Wheel Drive'),
  Text(5, 0, 'All-Wheel Drive'),
  Text(6, 0, '4-Wheel Drive'),
  Text(7, 0, 'Part-time 4-Wheel Drive'),
  Text(8, 0, 'Automatic (A1)')])
```

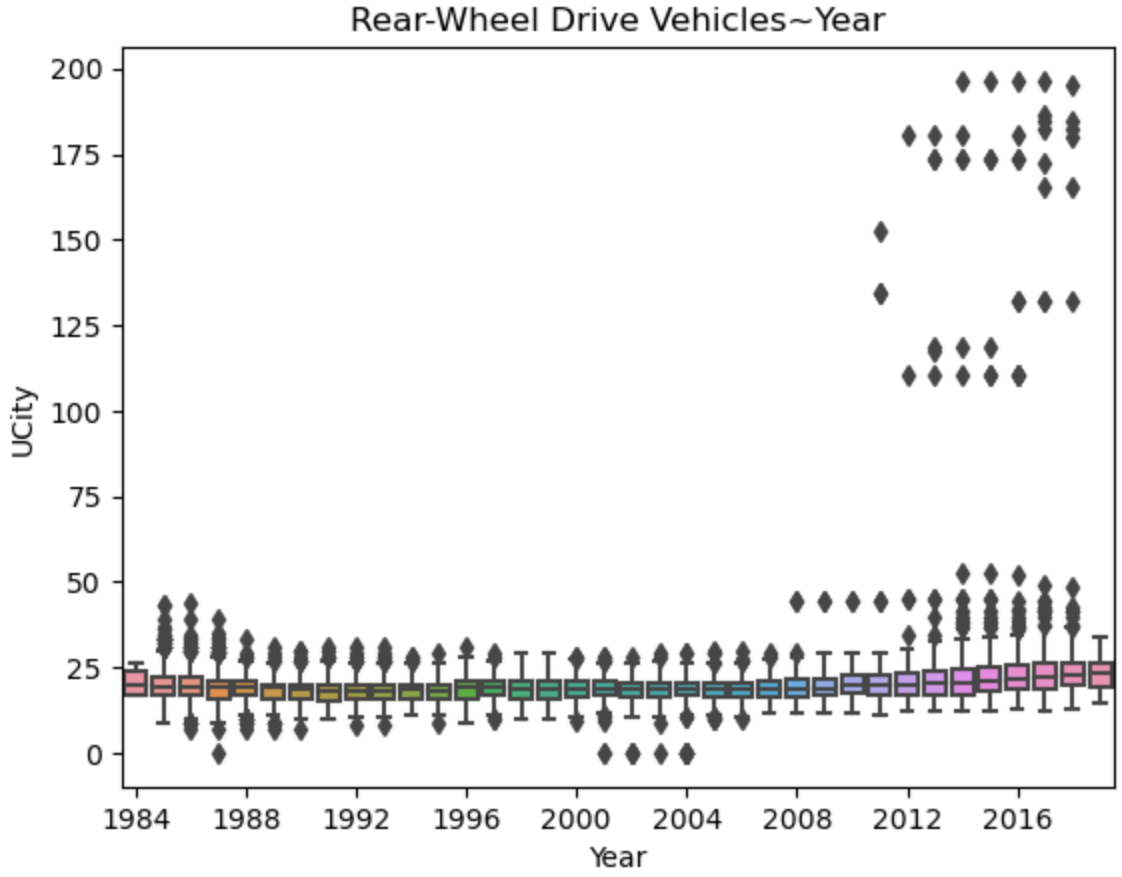


In [651…

```
# box plot Rear Wheel Vehicles
df_rw = df[df['drive'] == 'Rear-Wheel Drive']

ax = sns.boxplot(data = df_rw, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "Rear-Wheel Drive Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 4) == 0:
        x_tick_numbers.append(i)
```

```
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```

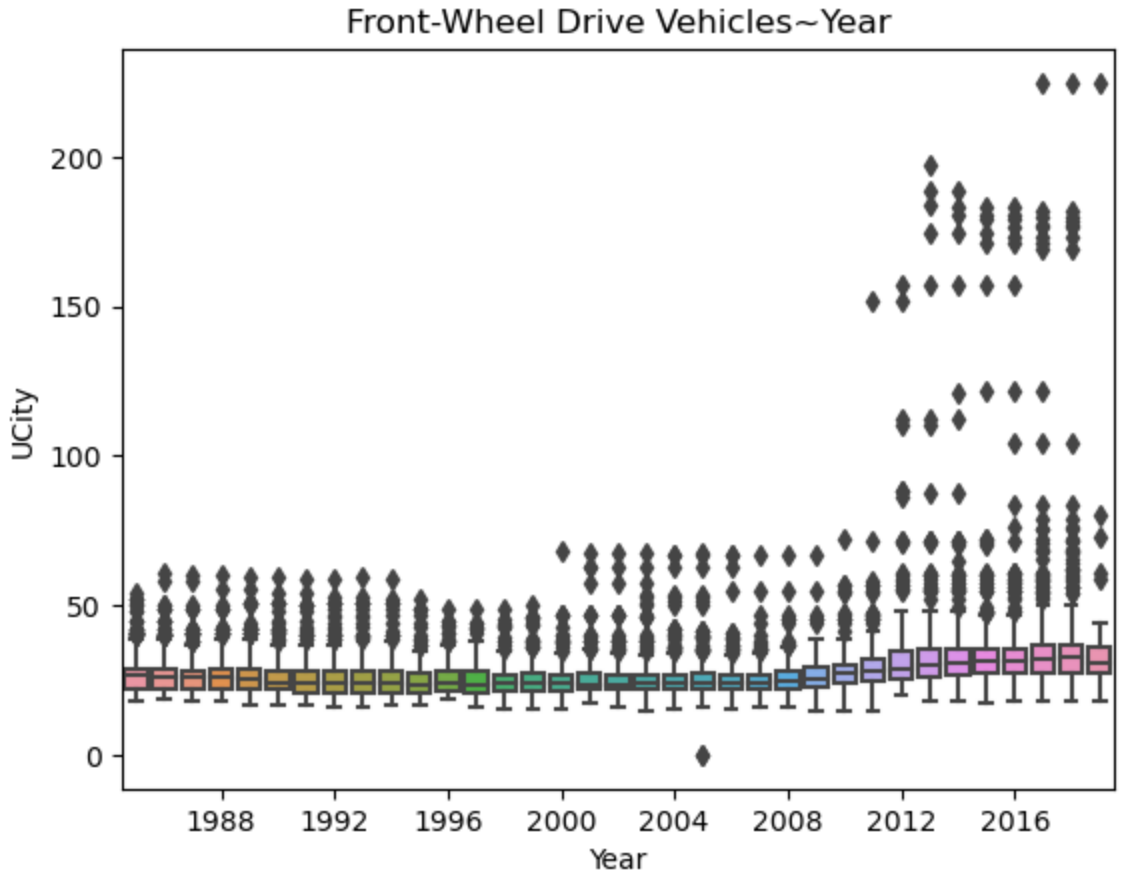


In [652…

```
# box plot Front Wheel Vehicles
df_fw = df[df['drive'] == 'Front-Wheel Drive']

ax = sns.boxplot(data = df_fw, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "Front-Wheel Drive Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 4) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992', '1993',
```

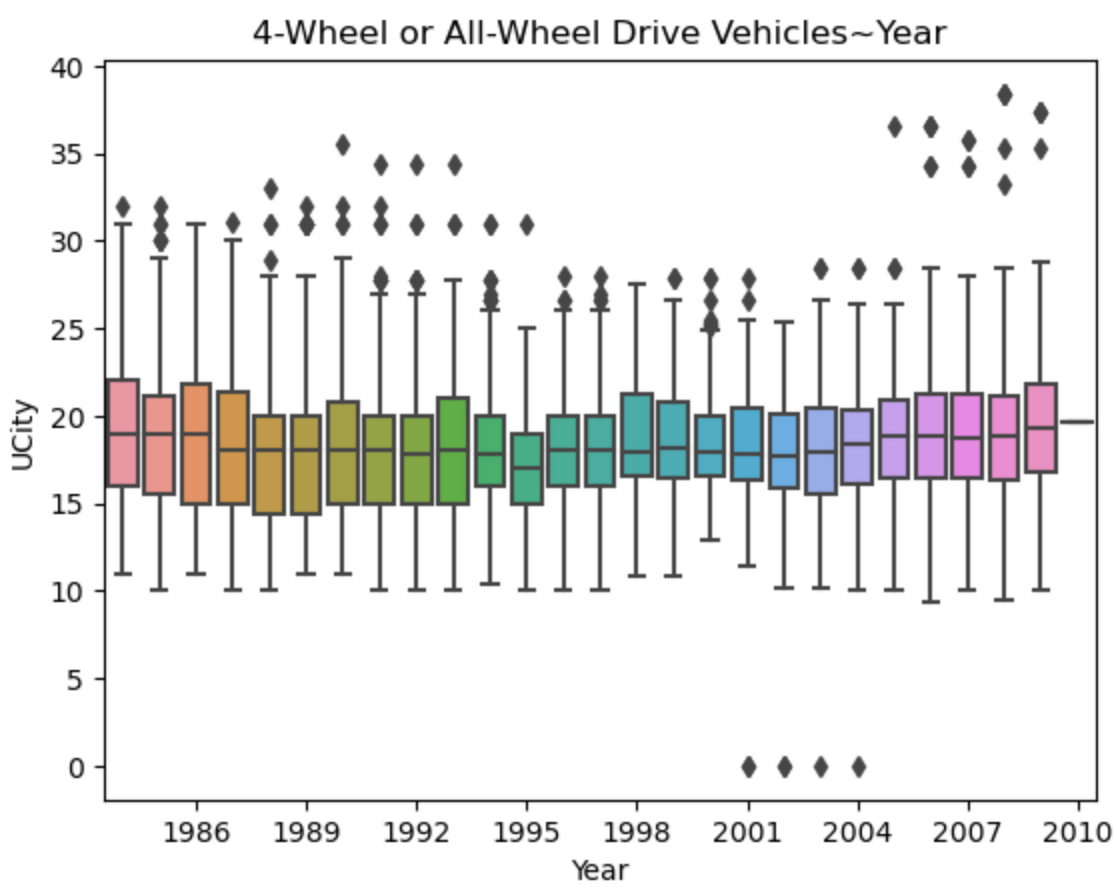
```
'1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003', '2004',
'2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014', '2015',
'2016', '2017', '2018', '2019']
```



In [653…

```
# box plot 4-Wheel or All Wheel Vehicles
df_aw = df[df['drive'] == '4-Wheel or All-Wheel Drive']

ax = sns.boxplot(data = df_aw, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "4-Wheel or All-Wheel Drive Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 3) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

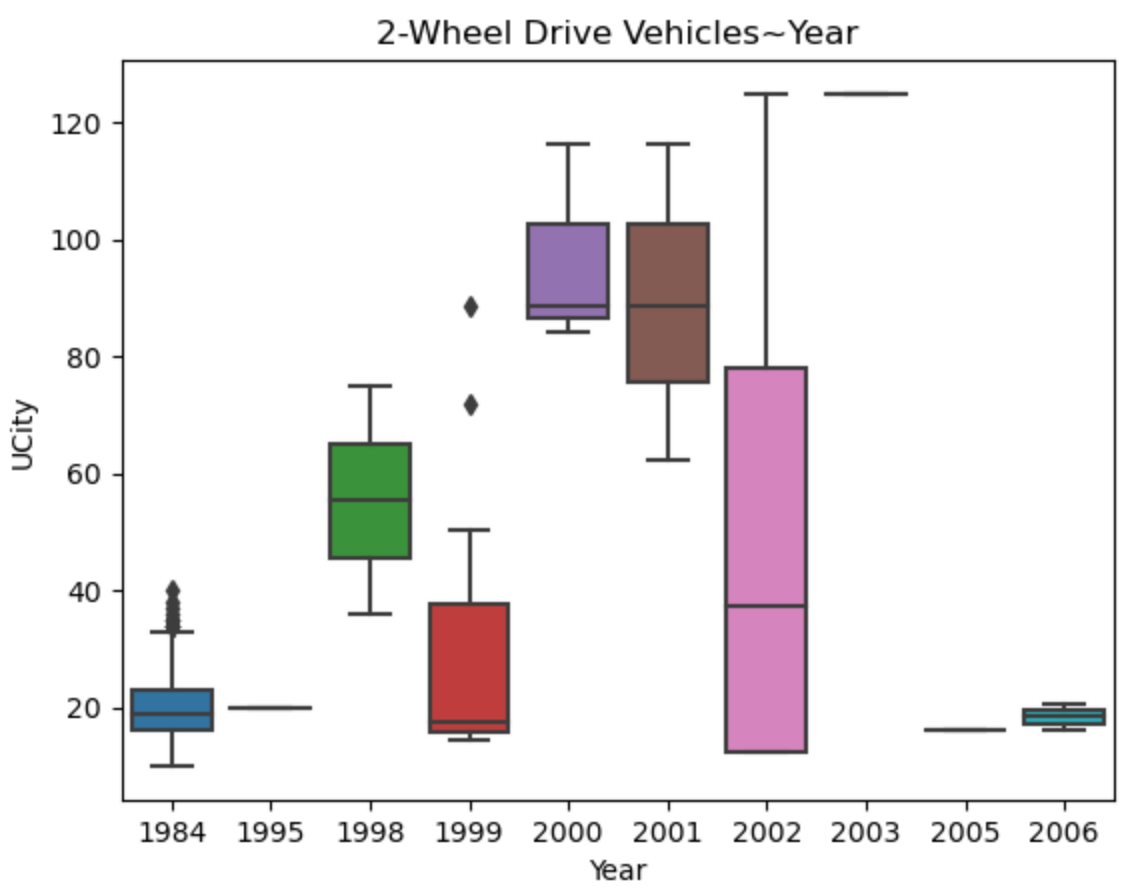
```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010']
```



In [654…

```
# box plot 2 Wheel Vehicles
df_2w = df[df['drive'] == '2-Wheel Drive']

ax = sns.boxplot(data = df_2w, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "2-Wheel Drive Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 1) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

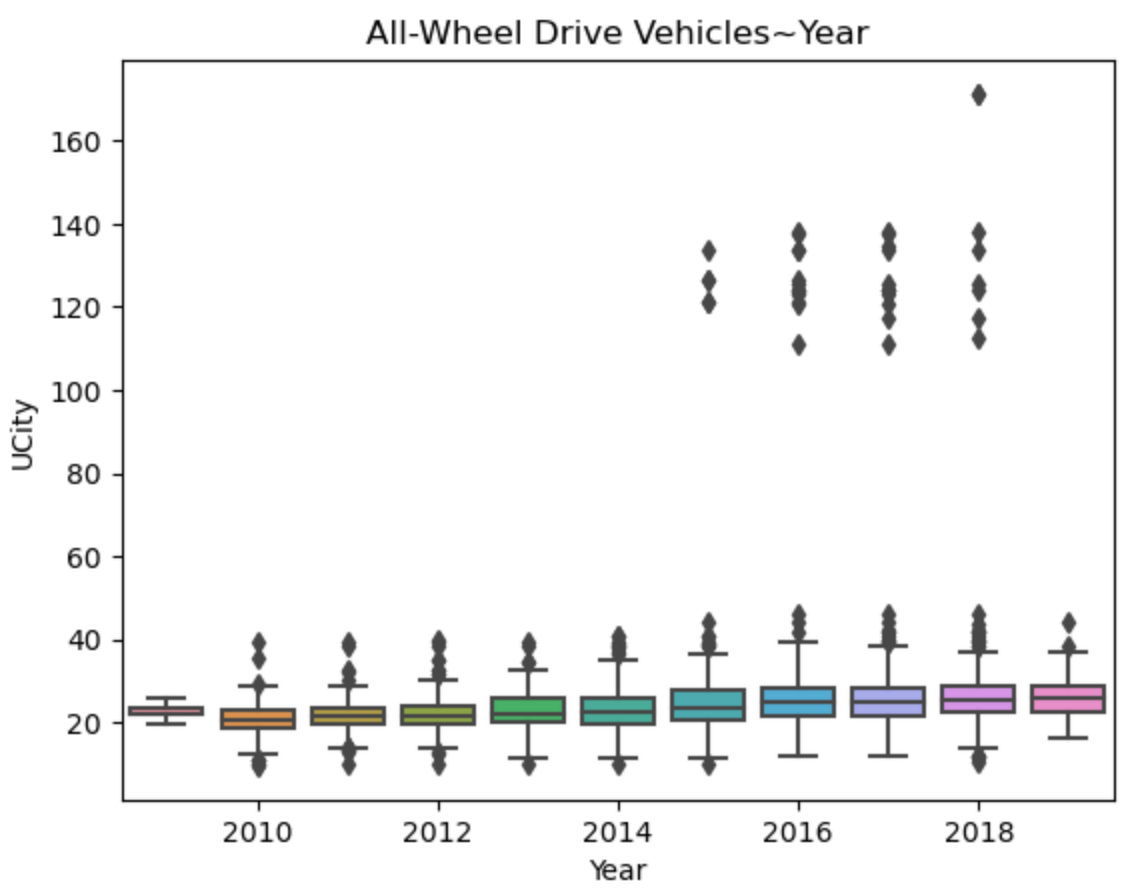
```
x_ticklabels: ['1984', '1995', '1998', '1999', '2000', '2001', '2002', '2003', '2005',
'2006']
```



In [655…

```
# box plot All-Wheel Vehicles
df_aw = df[df['drive'] == 'All-Wheel Drive']

ax = sns.boxplot(data = df_aw, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "All-Wheel Drive Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 2) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

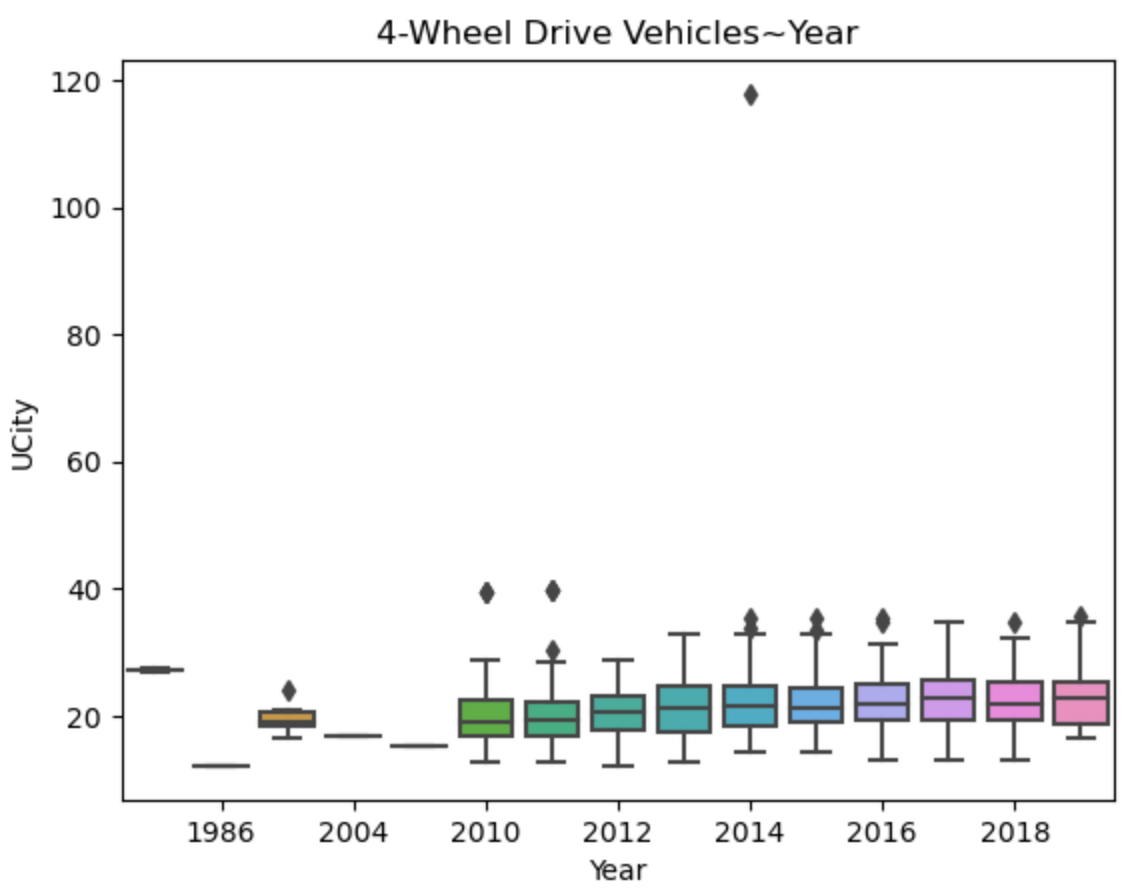
```
x_ticklabels: ['2009', '2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017',
'2018', '2019']
```



In [656…

```
# box plot 4-Wheel Vehicles
df_4w = df[df['drive'] == '4-Wheel Drive']

ax = sns.boxplot(data = df_4w, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "4-Wheel Drive Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 2) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

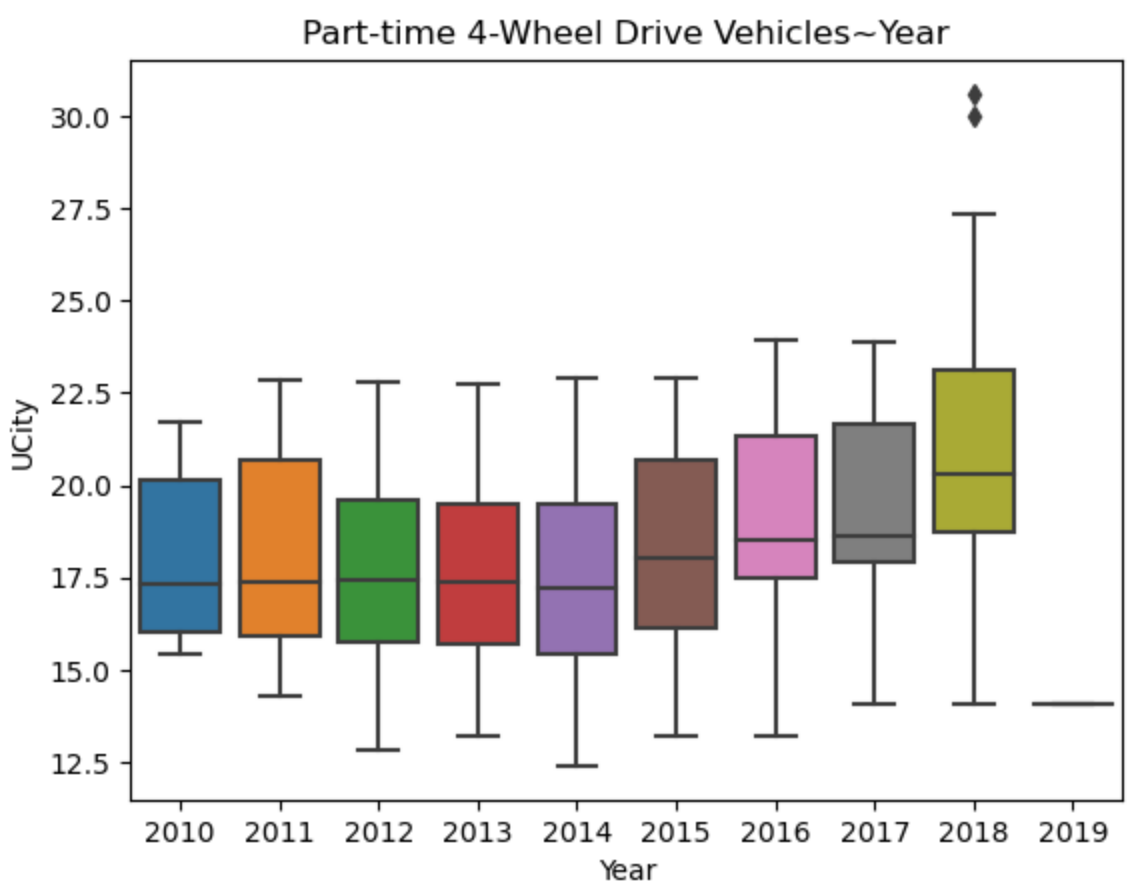
```
x_ticklabels: ['1985', '1986', '1997', '2004', '2007', '2010', '2011', '2012', '2013',
'2014', '2015', '2016', '2017', '2018', '2019']
```



In [657…

```
# box plot Part-time 4-Wheel Vehicles
df_p4w = df[df['drive'] == 'Part-time 4-Wheel Drive']

ax = sns.boxplot(data = df_p4w, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "Part-time 4-Wheel Drive Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 1) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

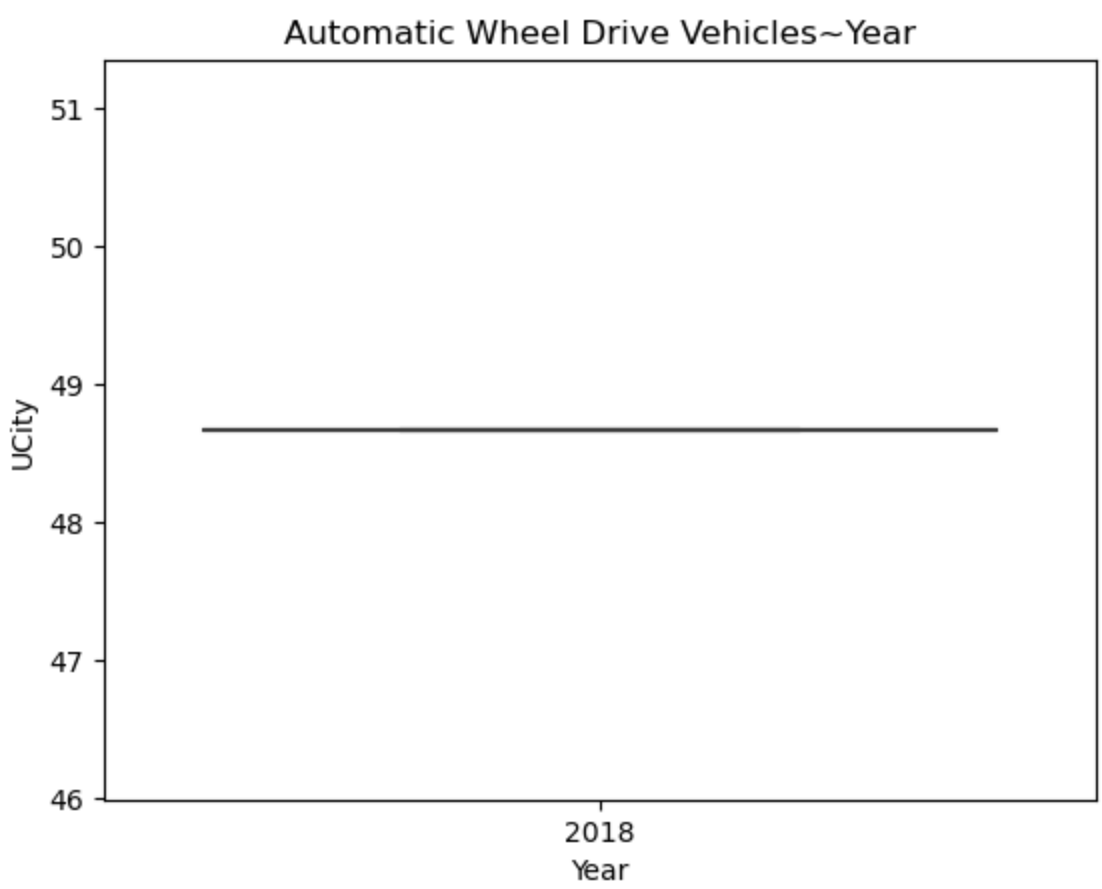
```
x_ticklabels: ['2010', '2011', '2012', '2013', '2014', '2015', '2016', '2017', '2018',
'2019']
```



In [658…

```
# box plot Automatic Wheel Vehicles
df_a = df[df['drive'] == 'Automatic (A1)']

ax = sns.boxplot(data = df_a, x = "year", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "Automatic Wheel Drive Vehicles~Year", xlabel = "Year")

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 1) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['2018']
```



Categorical + Logical - guzzler

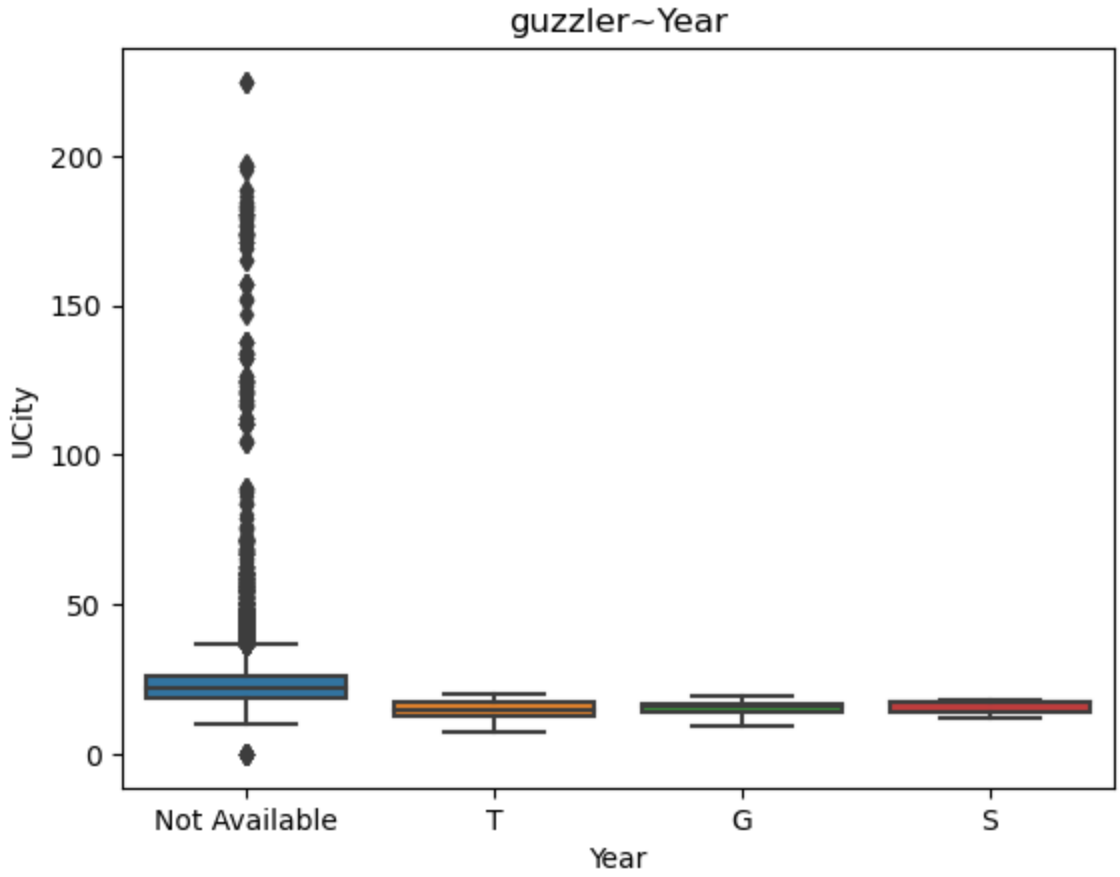
In [659…
```
df['guzzler'].describe()
```

Out[659]:
```
count              40081
unique                 4
top       Not Available
freq               37704
Name: guzzler, dtype: object
```

In [660…
```
df['guzzler'].value_counts()
```

Out[660]:
```
Not Available    37704
G                 1398
T                  964
S                   15
Name: guzzler, dtype: int64
```

In [661…
```
# box plot guzzler-UCity

ax = sns.boxplot(data = df, x = "guzzler", y = "UCity")

# set title and redefine the xlabel
ax.set(title = "guzzler~Year", xlabel = "Year")

plt.show()
```

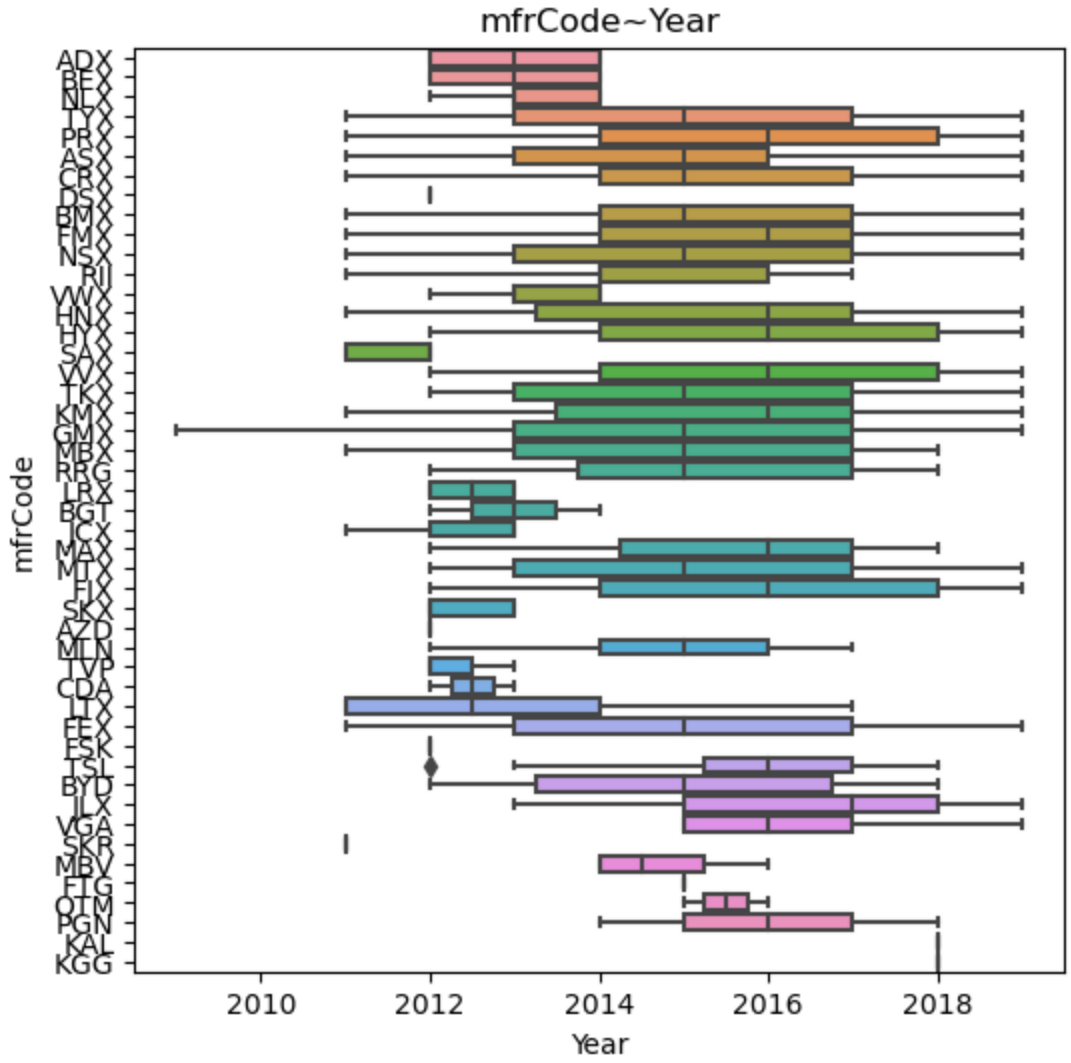


Categorical + Logical - mfrCode

In [662…
```
df['mfrCode'].describe()
```

Out[662]:
```
count              40081
unique                48
top       Not Available
freq              30818
Name: mfrCode, dtype: object
```

In [663…
```
# box plot Automatic Wheel Vehicles

plt.figure(figsize = (6, 6))
df_mf = df[df['mfrCode'] != 'Not Available']

ax = sns.boxplot(data = df_mf, x = "year", y = "mfrCode")

# set title and redefine the xlabel
ax.set(title = "mfrCode~Year", xlabel = "Year")

plt.show()
```



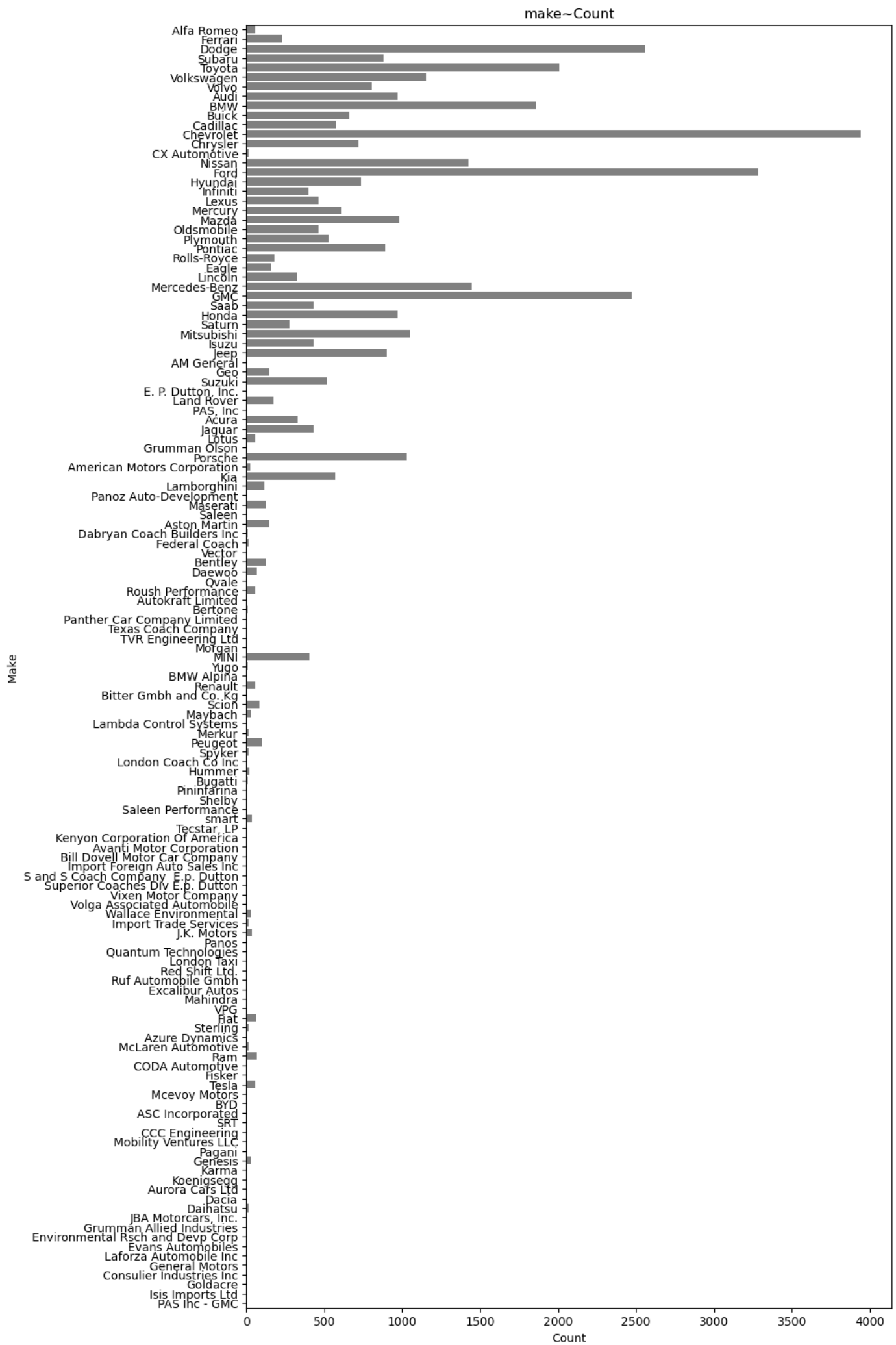
Categorical + Logical - make

In [664…
```
df['make'].describe()
```

Out[664]:
```
count          40081
unique           135
top        Chevrolet
freq            3944
Name: make, dtype: object
```

In [665…
```
# plot count graph for make

plt.figure(figsize = (10, 20))

ax = sns.countplot(y = df['make'], color = 'grey')
ax.set(title = "make~Count", xlabel = "Count", ylabel = "Make")
```

Out[665]:
```
[Text(0.5, 1.0, 'make~Count'), Text(0.5, 0, 'Count'), Text(0, 0.5, 'Make')]
```


make~Count
Make
Count
Alfa Romeo
Ferrari
Dodge
Subaru
Toyota
Volkswagen
Volvo
Audi
BMW
Buick
Cadillac
Chevrolet
Chrysler
CX Automotive
Nissan
Ford
Hyundai
Infiniti
Lexus
Mercury
Mazda
Oldsmobile
Plymouth
Pontiac
Rolls-Royce
Eagle
Lincoln
Mercedes-Benz
GMC
Saab
Honda
Saturn
Mitsubishi
Isuzu
Jeep
AM General
Geo
Suzuki
E. P. Dutton, Inc.
Land Rover
PAS, Inc
Acura
Jaguar
Lotus
Grumman Olson
Porsche
American Motors Corporation
Kia
Lamborghini
Panoz Auto-Development
Maserati
Saleen
Aston Martin
Dabryan Coach Builders Inc
Federal Coach
Vector
Bentley
Daewoo
Qvale
Roush Performance
Autokraft Limited
Bertone
Panther Car Company Limited
Texas Coach Company
TVR Engineering Ltd
Morgan
MINI
Yugo
BMW Alpina
Renault
Bitter Gmbh and Co. Kg
Scion
Maybach
Lambda Control Systems
Merkur
Peugeot
Spyker
London Coach Co Inc
Hummer
Bugatti
Pininfarina
Shelby
Saleen Performance
smart
Tecstar, LP
Kenyon Corporation Of America
Avanti Motor Corporation
Bill Dovell Motor Car Company
Import Foreign Auto Sales Inc
S and S Coach Company E.p. Dutton
Superior Coaches Div E.p. Dutton
Vixen Motor Company
Volga Associated Automobile
Wallace Environmental
Import Trade Services
J.K. Motors
Panos
Quantum Technologies
London Taxi
Red Shift Ltd.
Ruf Automobile Gmbh
Excalibur Autos
Mahindra
VPG
Fiat
Sterling
Azure Dynamics
McLaren Automotive
Ram
CODA Automotive
Fisker
Tesla
Mcevoy Motors
BYD
ASC Incorporated
SRT
CCC Engineering
Mobility Ventures LLC
Pagani
Genesis
Karma
Koenigsegg
Aurora Cars Ltd
Dacia
Daihatsu
JBA Motorcars, Inc.
Grumman Allied Industries
Environmental Rsch and Devp Corp
Evans Automobiles
Laforza Automobile Inc
General Motors
Consulier Industries Inc
Goldacre
Isis Imports Ltd
PAS Inc - GMC
0
500
1000
1500
2000
2500
3000
3500
4000

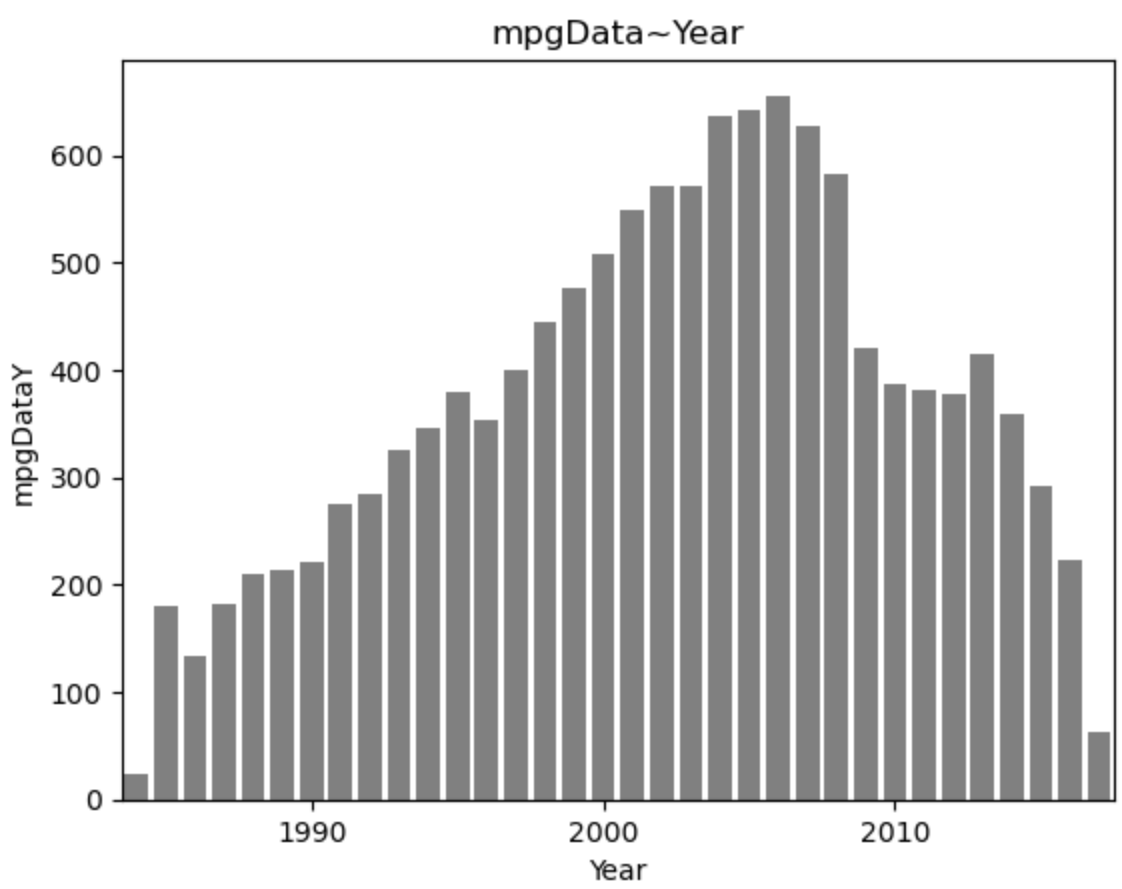
Categorical + Logical - mpgData

In [666…
```
df['mpgData'].describe()
```

Out[666]:
```
count     40081
unique        2
top           N
freq      27367
Name: mpgData, dtype: object
```

In [667…
```
# plot count mpgData
df_y = df[df['mpgData'] == 'Y']
ax = sns.countplot(data = df_y, x = "year", color = 'grey')

# set title and redefine the xlabel
ax.set(title = "mpgData~Year", xlabel = "Year", ylabel = 'mpgDataY')

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017']
```

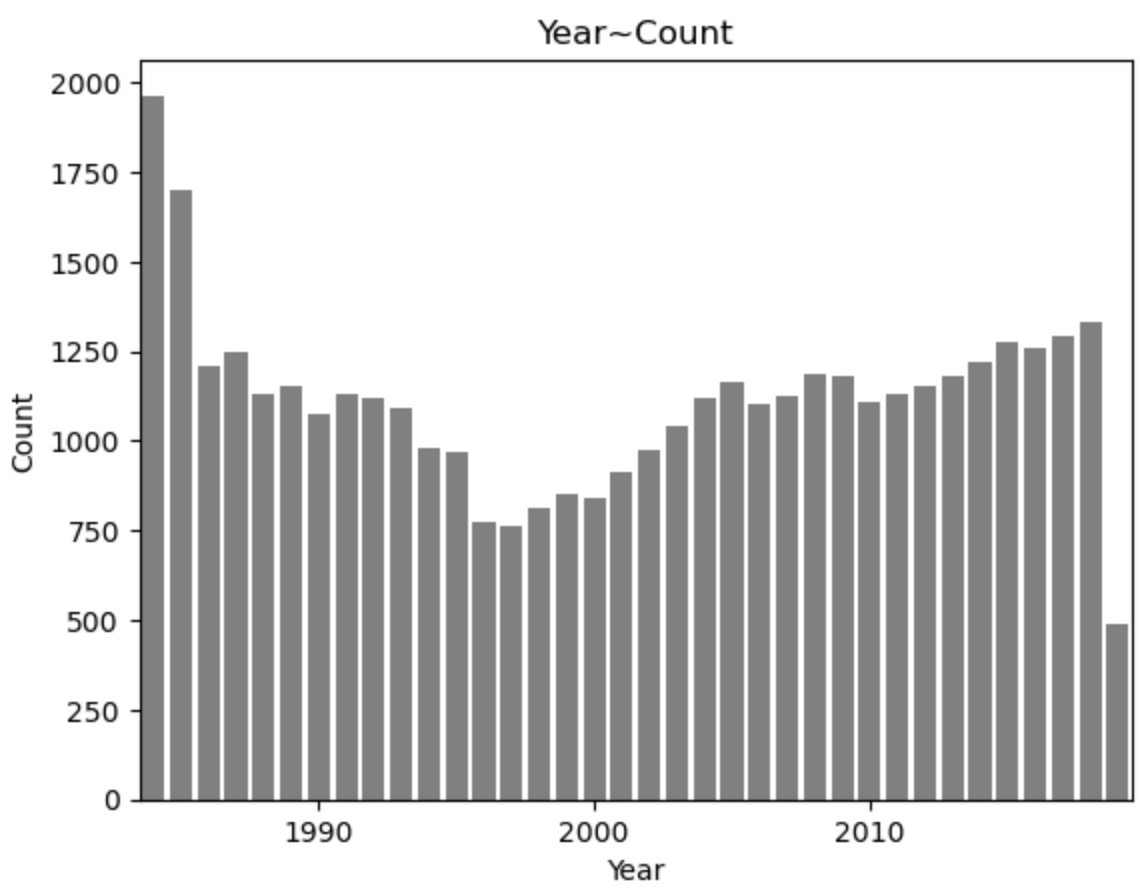


Categorical + Logical - year

In [668…

```
# plot count year
ax = sns.countplot(data = df, x = "year", color = 'grey')

# set title and redefine the xlabel
ax.set(title = "Year~Count", xlabel = "Year", ylabel = 'Count')

# get xtick labels
labels = [t.get_text() for t in ax.get_xticklabels()]

print("x_ticklabels:", labels)

# modify xtick labels to a condensed version to show at every 10
x_tick_numbers = []
x_tick_labels = []

for i in range(len(labels)):
    if (int(labels[i]) % 10) == 0:
        x_tick_numbers.append(i)
        x_tick_labels.append(labels[i])

# update xtick labels
plt.xticks(ticks = x_tick_numbers, labels = x_tick_labels)
plt.show()
```

```
x_ticklabels: ['1984', '1985', '1986', '1987', '1988', '1989', '1990', '1991', '1992',
'1993', '1994', '1995', '1996', '1997', '1998', '1999', '2000', '2001', '2002', '2003',
'2004', '2005', '2006', '2007', '2008', '2009', '2010', '2011', '2012', '2013', '2014',
'2015', '2016', '2017', '2018', '2019']
```

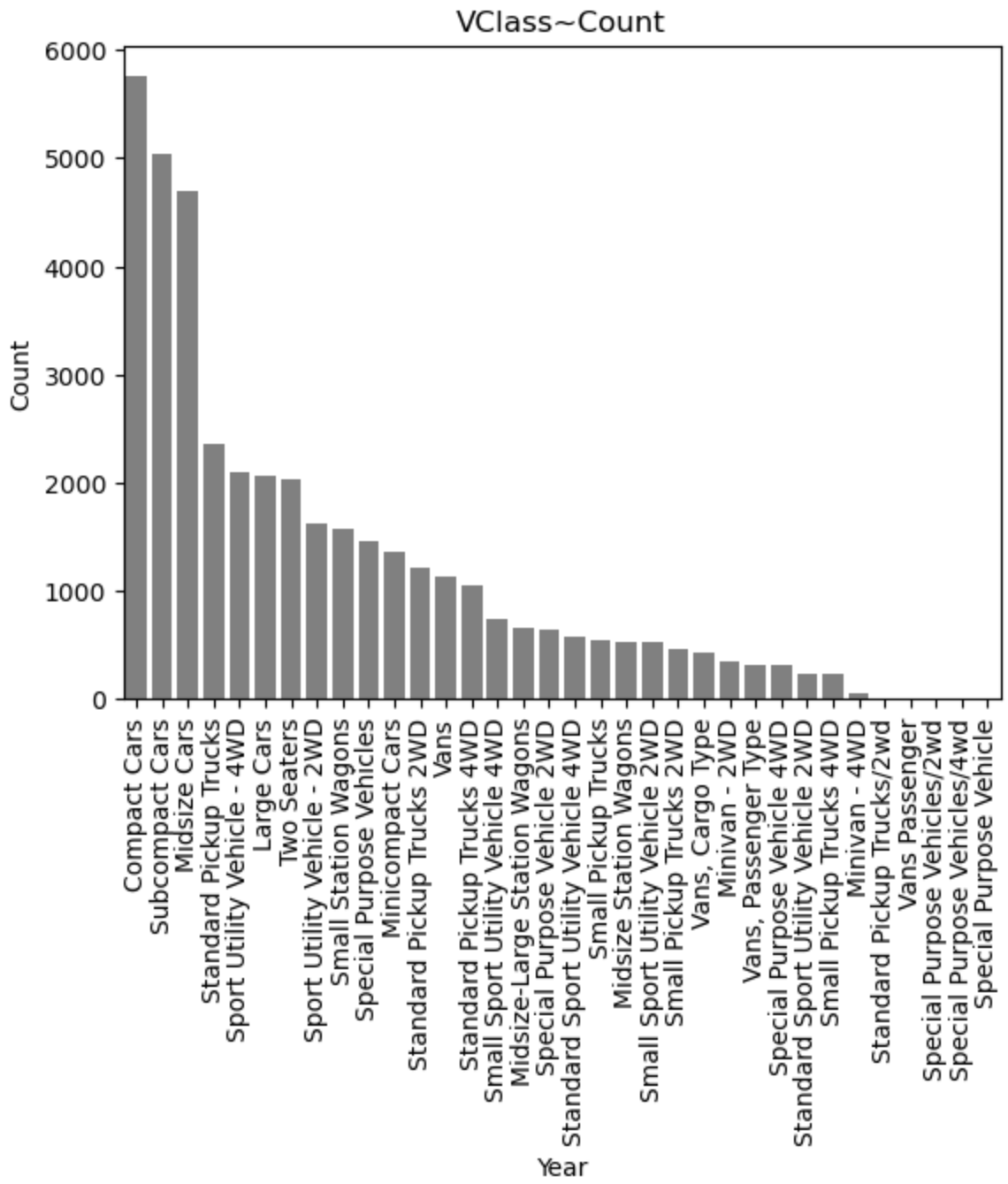


Categorical + Logical - VClass

In [669…
```
df['VClass'].describe()
```

Out[669]:
```
count             40081
unique               34
top       Compact Cars
freq              5751
Name: VClass, dtype: object
```

In [670…
```
# plot count year
ax = sns.countplot(data = df, x = "VClass", color = 'grey', order = df['VClass'].value_c

# set title and redefine the xlabel
ax.set(title = "VClass~Count", xlabel = "Year", ylabel = 'Count')

plt.xticks(rotation = 90)
plt.show()
```

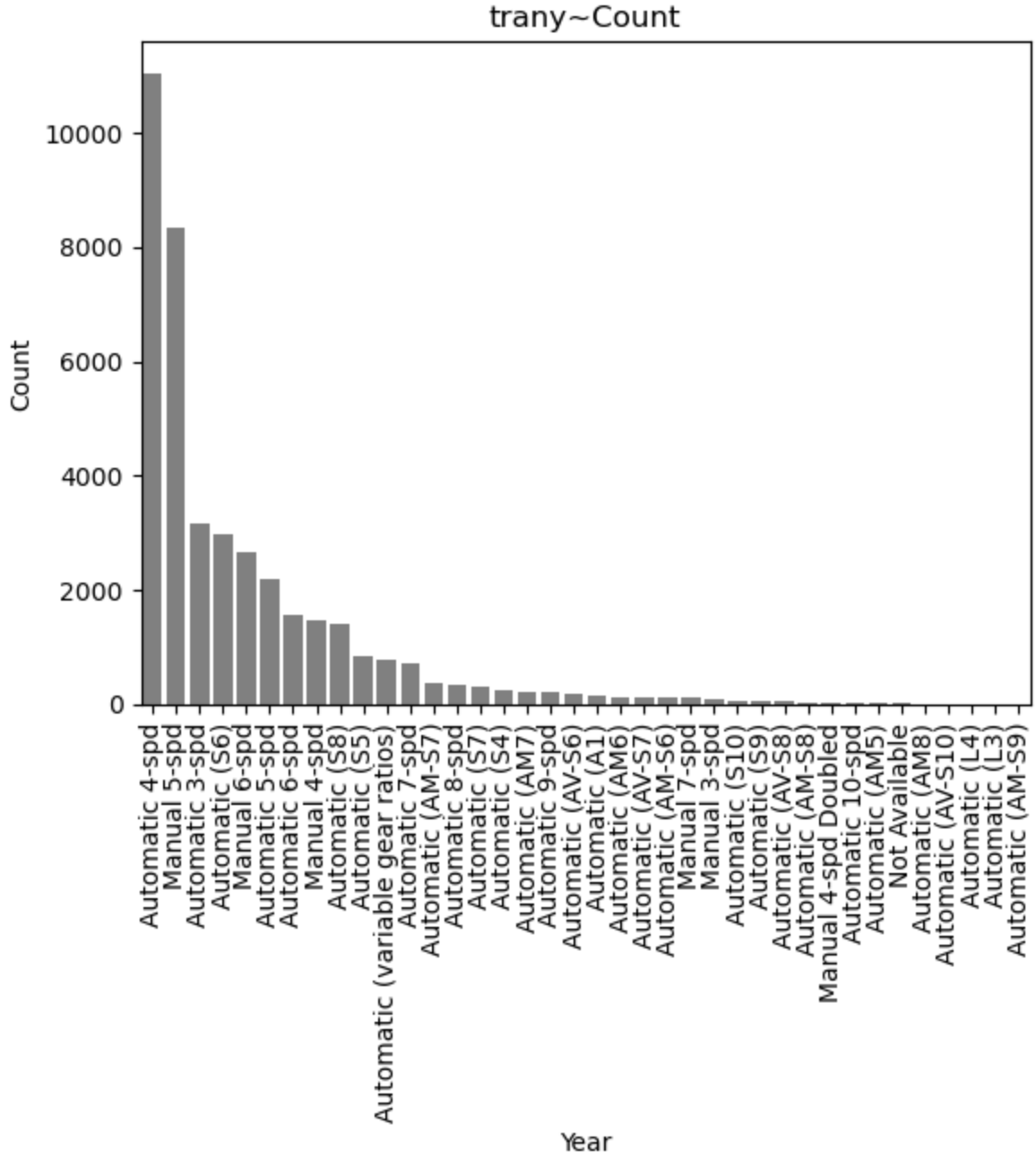


Categorical + Logical - trany

In [671…
```
df['trany'].describe()
```

Out[671]:
```
count                40081
unique                  38
top       Automatic 4-spd
freq                 11045
Name: trany, dtype: object
```

In [672…
```
# plot count year
ax = sns.countplot(data = df, x = "trany", color = 'grey', order = df['trany'].value_cou

# set title and redefine the xlabel
ax.set(title = "trany~Count", xlabel = "Year", ylabel = 'Count')

plt.xticks(rotation = 90)
plt.show()
```

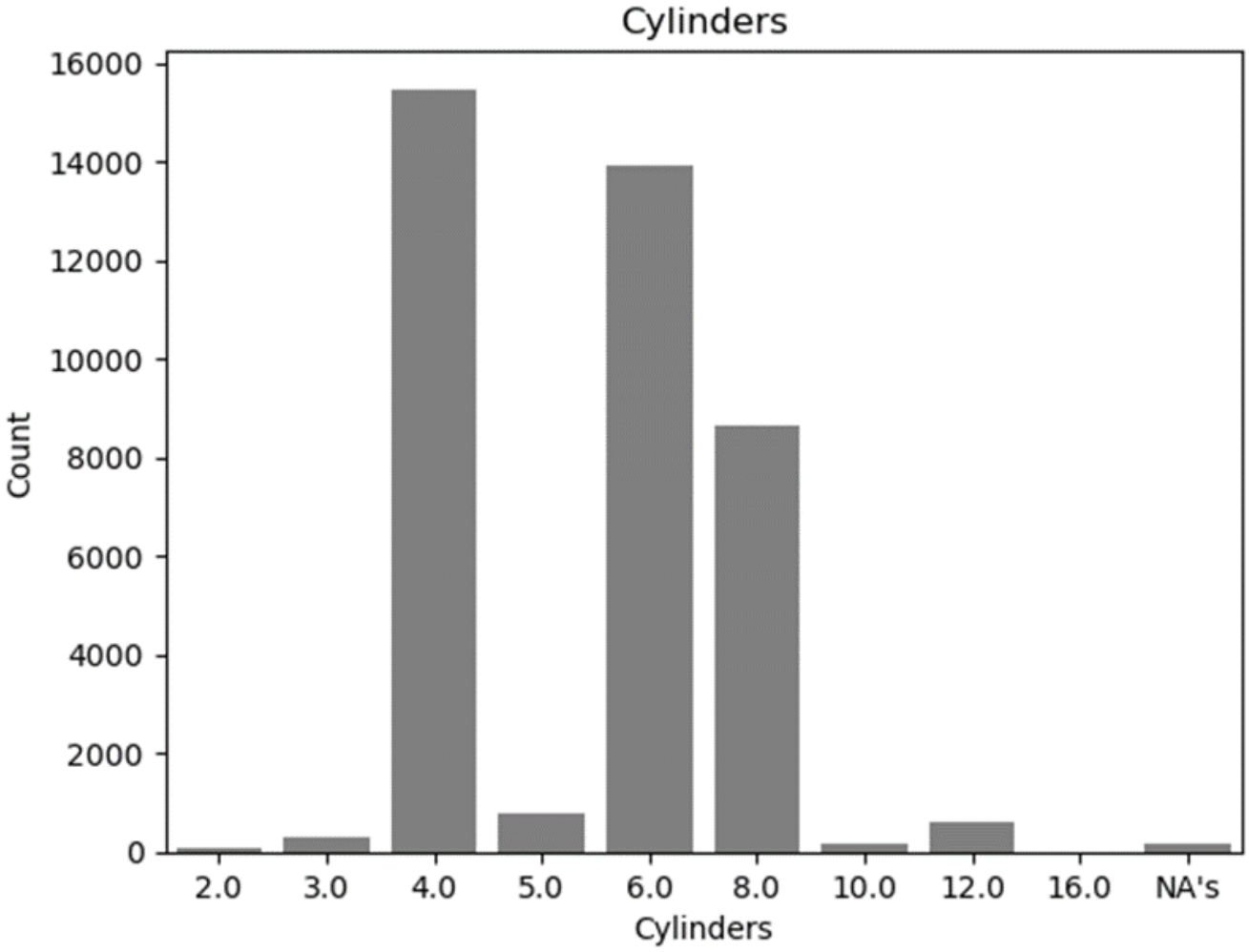


Theory Questions:

1.

In replicating the data analysis as provided in the specimen and to enhance the reliability of plots and correlations, the data was cleaned that involved excluding the values that were not available. For instance, consider the numeric variable atvType, which represents the type of alternative fuel or advanced technology vehicle. In this case, 38248 (91%) of the vehicles were categorized as “Not Available”. These values were removed to observe the influence of various fuel types on vehicles over period.

An interesting aspect that was observed during the data analysis revealed a negative correlation between cylinders and UCity, unadjusted MPG. The observed -0.67 correlation underscores that the bigger vehicles tend to achieve fewer miles per gallon, highlighting an inverse relationship between number of cylinders and fuel efficiency. However, when compared different cylinders over period, vehicles with 4, 6, and 8 cylinders were getting slightly better on UCity.



2.

The choice for a regression model is based on the continuous nature of the dependent variable, UCity. Regression models are preferred in this given context because they are designed to predict numeric values, making them well suited for these scenarios where the outcome is a quantity that can vary over a range.

3.

1. The variables that shouldn't be considered as inputs to the model are:

   a. chargeXXXX: Data is absent for all three charge variables—charge120, charge240, and charge240b, and no insights were provided with UCity.

   b. range: A total of 39913 vehicles don't have any range information. While the remaining 168 vehicles have a strong correlation of 0.60 with UCity, it isn't reliable because the distribution is heteroskedastic.

   c. engId: It is just an index variable and has no use for prediction.

   d. evMotor: A total of 38345 vehicles don't have any evMotor information, and only one specific motor is outperforming UCity measure. Thus, this variable in the model can lead to overfitting.

4.

To avoid underfitting/overfitting variables such as evMotor should be avoided because one specific motor is outperforming UCity measure, and this could lead to overfitting for the model. Additionally, to evaluate the model on underfitting/overfitting a few techniques can be employed such as train-test split, cross validation, early stopping, feature engineering, etc.

In [ ]: