In [45]:
```
'''
Question 1:
For the MTCARS dataset, answer the specified questions with summarization an
'''
import pandas as pd
import matplotlib.pyplot as plt
dat_mtcars = pd.read_csv("mtcars.csv",index_col=0)
dat_mtcars
```

Out[45]:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
model											
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0	1	4	4
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0	1	4	4
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1	1	4	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1	0	3	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0	0	3	2
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1	0	3	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0	0	3	4
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1	0	4	2
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1	0	4	2
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1	0	4	4
Merc 280C	17.8	6	167.6	123	3.92	3.440	18.90	1	0	4	4
Merc 450SE	16.4	8	275.8	180	3.07	4.070	17.40	0	0	3	3
Merc 450SL	17.3	8	275.8	180	3.07	3.730	17.60	0	0	3	3
Merc 450SLC	15.2	8	275.8	180	3.07	3.780	18.00	0	0	3	3
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4
Chrysler Imperial	14.7	8	440.0	230	3.23	5.345	17.42	0	0	3	4
Fiat 128	32.4	4	78.7	66	4.08	2.200	19.47	1	1	4	1
Honda Civic	30.4	4	75.7	52	4.93	1.615	18.52	1	1	4	2
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.90	1	1	4	1
Toyota Corona	21.5	4	120.1	97	3.70	2.465	20.01	1	0	3	1
Dodge Challenger	15.5	8	318.0	150	2.76	3.520	16.87	0	0	3	2
AMC Javelin	15.2	8	304.0	150	3.15	3.435	17.30	0	0	3	2
Camaro Z28	13.3	8	350.0	245	3.73	3.840	15.41	0	0	3	4
Pontiac Firebird	19.2	8	400.0	175	3.08	3.845	17.05	0	0	3	2
Fiat X1-9	27.3	4	79.0	66	4.08	1.935	18.90	1	1	4	1
Porsche 914-2	26.0	4	120.3	91	4.43	2.140	16.70	0	1	5	2
Lotus Europa	30.4	4	95.1	113	3.77	1.513	16.90	1	1	5	2
Ford Pantera L	15.8	8	351.0	264	4.22	3.170	14.50	0	1	5	4
Ferrari Dino	19.7	6	145.0	175	3.62	2.770	15.50	0	1	5	6
Maserati Bora	15.0	8	301.0	335	3.54	3.570	14.60	0	1	5	8
Volvo 142E	21.4	4	121.0	109	4.11	2.780	18.60	1	1	4	2

In [3]:
```
dat_mtcars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 32 entries, Mazda RX4 to Volvo 142E
Data columns (total 11 columns):
 #   Column  Non-Null Count  Dtype  
---  ------  --------------  -----  
 0   mpg     32 non-null     float64
 1   cyl     32 non-null     int64  
 2   disp    32 non-null     float64
 3   hp      32 non-null     int64  
 4   drat    32 non-null     float64
 5   wt      32 non-null     float64
 6   qsec    32 non-null     float64
 7   vs      32 non-null     int64  
 8   am      32 non-null     int64  
 9   gear    32 non-null     int64  
 10  carb    32 non-null     int64  
dtypes: float64(5), int64(6)
memory usage: 3.0+ KB
```

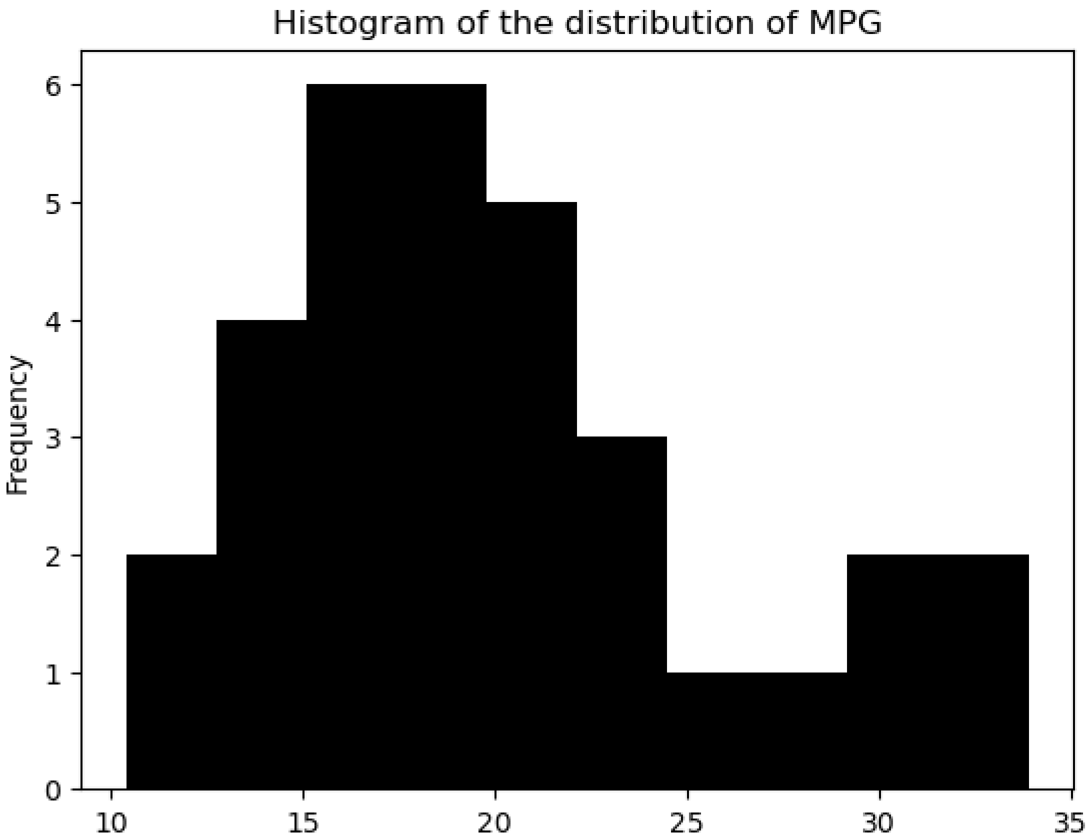
In [4]:
```
dat_mtcars.describe()
```

Out[4]:

	mpg	cyl	disp	hp	drat	wt	qsec	
count	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000000	32.000
mean	20.090625	6.187500	230.721875	146.687500	3.596563	3.217250	17.848750	0.437
std	6.026948	1.785922	123.938694	68.562868	0.534679	0.978457	1.786943	0.504
min	10.400000	4.000000	71.100000	52.000000	2.760000	1.513000	14.500000	0.000
25%	15.425000	4.000000	120.825000	96.500000	3.080000	2.581250	16.892500	0.000
50%	19.200000	6.000000	196.300000	123.000000	3.695000	3.325000	17.710000	0.000
75%	22.800000	8.000000	326.000000	180.000000	3.920000	3.610000	18.900000	1.000
max	33.900000	8.000000	472.000000	335.000000	4.930000	5.424000	22.900000	1.000

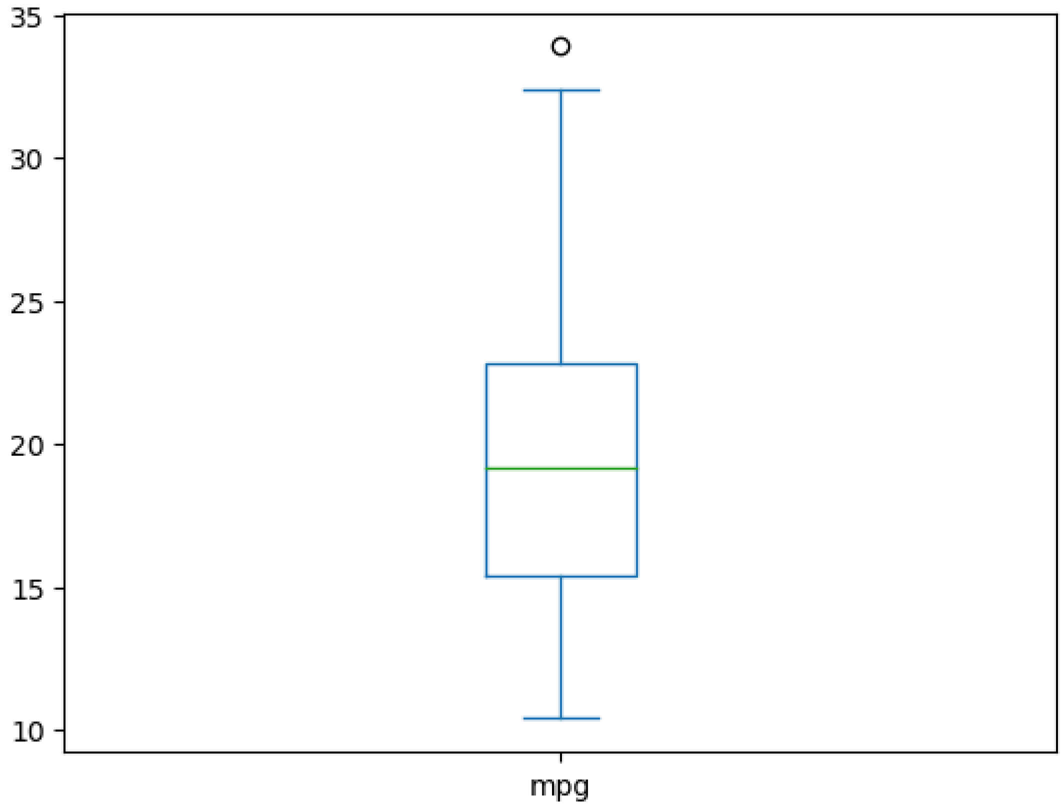
In [34]:
```
# 1. Explore the distribution of fuel efficiency of the cars.
# Remember to label both the axes and put a title to the plot.
dat_mtcars["mpg"].plot(kind = 'hist',x='mpg',y='frequency',bins=10, title="H
```

Out[34]:
```
<AxesSubplot:title={'center':'Histogram of the distribution of MPG'}, ylab
el='Frequency'>
```



In [22]:
```
dat_mtcars["mpg"].plot(kind="box")
```

Out[22]: <AxesSubplot:>



In [27]:
```
# 2.1 Cars with best efficiency.
max_mpg=dat_mtcars["mpg"].max()
dat_mtcars[dat_mtcars["mpg"]==max_mpg]
```

Out[27]:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
model											
Toyota Corolla	33.9	4	71.1	65	4.22	1.835	19.9	1	1	4	1

In [26]:
```
# 2.2 Cars with worst fuel efficiency.
min_mpg=dat_mtcars["mpg"].min()
dat_mtcars[dat_mtcars["mpg"]==min_mpg]
```

Out[26]:

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
model											
Cadillac Fleetwood	10.4	8	472.0	205	2.93	5.250	17.98	0	0	3	4
Lincoln Continental	10.4	8	460.0	215	3.00	5.424	17.82	0	0	3	4

In [18]:
```
# Features like am, gearsw are numeric but binary features. They are used to
# 3. How many cars are manual v/s automatic?
dat_mtcars["am"].nunique()
dat_mtcars["am"].value_counts()
```

Out[18]:
```
0    19
1    13
Name: am, dtype: int64
```

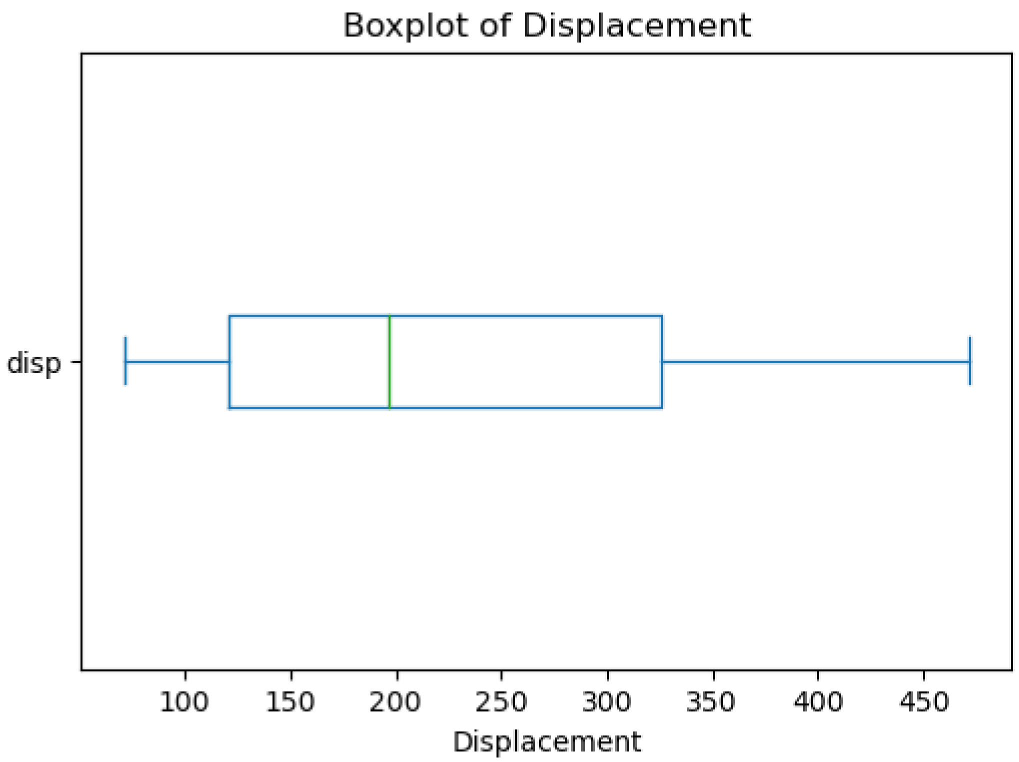
In [38]:
```
dat_mtcars=pd.read_csv("mtcars.csv")
```

In [39]:
```
# 4. Car with the worst horsepower
worst_hp_car = dat_mtcars[dat_mtcars['hp'] == dat_mtcars['hp'].min()][['mode
print(f"Car with worst horsepower:\n{worst_hp_car}")
```

```
Car with worst horsepower:
          model  hp
18  Honda Civic  52
```

In [47]:
```
# 5. Find 5 number summary and draw boxplot of displacement.
disp_summary=dat_mtcars['disp'].describe()[['min','25%','50%','75%','max']]
print(f"5 number summary of displacement:\n{disp_summary}")
dat_mtcars['disp'].plot(kind='box', vert=False, figsize=(6, 4), title='Boxpl
plt.xlabel("Displacement")
plt.show()
```

```
5 number summary of displacement:
min     71.100
25%    120.825
50%    196.300
75%    326.000
max    472.000
Name: disp, dtype: float64
```



In [50]:
```
dat_mtcars=pd.read_csv("mtcars.csv")
# 6. Which is the heaviest car? How many gears does it have?
heaviest_car = dat_mtcars[dat_mtcars['wt'] == dat_mtcars['wt'].max()][['mode
print(f"Heaviest car and its gears:\n{heaviest_car}")
```

```
Heaviest car and its gears:
                  model     wt  gear
15  Lincoln Continental  5.424     3
```

In [52]:
```python
# 7. Which is the car with the best qsec?
best_qsec_car = dat_mtcars[dat_mtcars['qsec'] == dat_mtcars['qsec'].min()][[
print(f"Car with best (fastest) qsec:\n{best_qsec_car}")
```

```
Car with best (fastest) qsec:
             model  qsec
28  Ford Pantera L  14.5
```

In [53]:
```python
# 8. What is average MPG for manual vs. automatic cars?
dat_man= dat_mtcars[dat_mtcars["am"]==1]["mpg"] # Manual cars are marked as
print("Average MPG of manual cars:",dat_man.mean())
dat_auto= dat_mtcars[dat_mtcars["am"]==0]["mpg"]
print("Average MPG of automatic cars:",dat_auto.mean())
```

```
Average MPG of manual cars: 24.39230769230769
Average MPG of automatic cars: 17.147368421052633
```

In [55]:
```python
# 9. Draw Side by Side box plot to understand the difference in fuel
# efficience of Manual vs Automatic cars. Analyze and write about fuel
# efficiency in each group (manual vs. automatic).
dat_mtcars = pd.read_csv("mtcars.csv")
fuel_efficiency_summary = dat_mtcars.groupby('am')['mpg'].describe()
print("\nFuel Efficiency Summary (MPG) for Manual vs Automatic Cars:")
print(fuel_efficiency_summary)
manual_mpg_mean = dat_mtcars[dat_mtcars['am'] == 'Manual']['mpg'].mean()
automatic_mpg_mean = dat_mtcars[dat_mtcars['am'] == 'Automatic']['mpg'].mear
print("\nFuel Efficiency Analysis:")
print(f"Manual Cars - Average MPG: {manual_mpg_mean:.2f}")
print(f"Automatic Cars - Average MPG: {automatic_mpg_mean:.2f}")
if manual_mpg_mean > automatic_mpg_mean:
    print("Manual cars tend to be more fuel efficient than Automatic cars.")
else:
    print("Automatic cars tend to be more fuel efficient than Manual cars.")
```

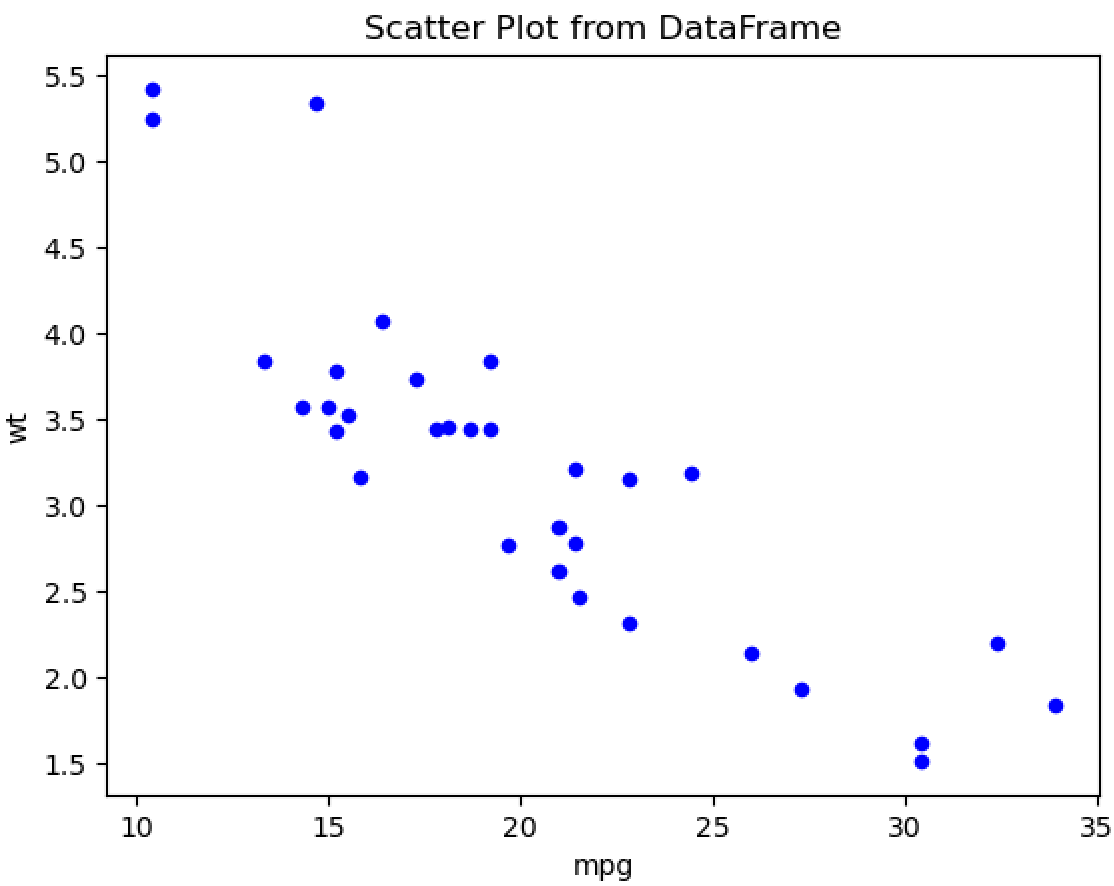
```

Fuel Efficiency Summary (MPG) for Manual vs Automatic Cars:
    count       mean       std   min    25%   50%   75%   max
am
0    19.0  17.147368  3.833966  10.4  14.95  17.3  19.2  24.4
1    13.0  24.392308  6.166504  15.0  21.00  22.8  30.4  33.9

Fuel Efficiency Analysis:
Manual Cars - Average MPG: nan
Automatic Cars - Average MPG: nan
Automatic cars tend to be more fuel efficient than Manual cars.
```

In [56]:
```
# 10. What is the relationship between the weight of the car and MPG?
dat_x= dat_mtcars[["mpg","wt"]]
dat_x= dat_x.reset_index(drop=True)
dat_x.plot(kind='scatter', x='mpg', y='wt', color='blue',
title='Scatter Plot from DataFrame')
```

Out[56]: <AxesSubplot:title={'center':'Scatter Plot from DataFrame'}, xlabel='mpg', ylabel='wt'>



In [58]:
```
# 11. Categorize the cars based on the number of gears in the cars.
# How many cars are there in each type?
dat_mtcars = pd.read_csv("mtcars.csv")
gear_counts = dat_mtcars['gear'].value_counts()
print(f"Number of cars based on gear type:\n{gear_counts}")
```

```
Number of cars based on gear type:
3    15
4    12
5     5
Name: gear, dtype: int64
```

In [60]:
```
# 12. What is the relationship between fuel efficiency and the number
# of gears in the car?
# Analyze the relationship between fuel efficiency and the number of gears.
gear_mpg_relation = dat_mtcars.groupby('gear')['mpg'].mean()
print(f"Average MPG for each gear type:\n{gear_mpg_relation}")
print('''\nAnalysis:\nHigher gear cars generally tend to have lower or simil
efficiency compared to lower gear cars, depending on the engine
performance and aerodynamics.''')
```

```
Average MPG for each gear type:
gear
3    16.106667
4    24.533333
5    21.380000
Name: mpg, dtype: float64

Analysis:
Higher gear cars generally tend to have lower or similar fuel
efficiency compared to lower gear cars, depending on the engine
performance and aerodynamics.
```

In [62]:
```
# 13. Explain the relationship between horse power and number of
# cylinders in the car.
hp_cyl_relation = dat_mtcars.groupby('cyl')['hp'].mean()
print(f"Average Horsepower for each Cylinder type:\n{hp_cyl_relation}")
print('''\nAnalysis:\nCars with more cylinders have higher horsepower since
generally have larger engines capable of producing more power.''')
```

```
Average Horsepower for each Cylinder type:
cyl
4     82.636364
6    122.285714
8    209.214286
Name: hp, dtype: float64

Analysis:
Cars with more cylinders have higher horsepower since they
generally have larger engines capable of producing more power.
```

In [63]:
```
# 14. Explain the relationship between displacement and gross horse power.
disp_hp_relation = dat_mtcars[['disp', 'hp']].corr()
print(f"Correlation between Displacement and Horsepower:\n{disp_hp_relation}
print('''\nAnalysis:\nLarger displacement engines usually generate more powe
this can also lead to increased fuel consumption.''')
```

```
Correlation between Displacement and Horsepower:
          disp        hp
disp  1.000000  0.790949
hp    0.790949  1.000000

Analysis:
Larger displacement engines usually generate more power, but
this can also lead to increased fuel consumption.
```

In [66]:
```
# 15. Which car would I pick if I am looking for high speed as well as
# good fuel efficiency?
dat_mtcars['performance_score'] = dat_mtcars['mpg'] / dat_mtcars['qsec']
best_car = dat_mtcars.loc[dat_mtcars['performance_score'].idxmax(), ['model'
'qsec']]
print(f"Best car for high speed and good fuel efficiency:\n{best_car}")
print("-----------------------------------------------------------------------
```

```
Best car for high speed and good fuel efficiency:
model     Lotus Europa
mpg               30.4
qsec              16.9
Name: 27, dtype: object
----------------------------------------------------------------------
```

In [1]:
```
'''
Question 2:
For the CEREALS dataset, answer the specified questions with summarization
and  effective visuals.
'''
import pandas as pd
df1 = pd.read_excel("Cereals.xls")
df1.replace(-1, pd.NA, inplace=True)
df1.to_csv("Cleaned_Cereals.csv", index=False)
df=pd.read_csv("Cleaned_Cereals.csv")
numeric_cols = ['rating', 'fiber', 'sugars', 'protein', 'calories']
df[numeric_cols] = df[numeric_cols].apply(pd.to_numeric,errors='coerce')
```

In [2]:
```
# 1. How many unique cereal brands are there?
unique_brands = df['name'].nunique()
print("Unique cereal brands:", unique_brands)
```

```
Unique cereal brands: 76
```

In [3]:
```
# 2. Number of cereals per manufacturer
cereals_per_manufacturer = df['mfr'].value_counts()
print(f"Cereals per manufacturer:\n{cereals_per_manufacturer}")
```

```
Cereals per manufacturer:
K    23
G    22
P     9
Q     8
R     8
N     5
A     1
Name: mfr, dtype: int64
```

In [4]:
```
# 3. Count of hot vs cold cereals
cereal_types = df['type'].value_counts()
print(f"Count of hot vs cold cereals:\n{cereal_types}")
```

```
Count of hot vs cold cereals:
C    73
H     3
Name: type, dtype: int64
```

In [5]:

```
# 4. Best and worst cereal based on rating
best_cereal = df.loc[df['rating'].idxmax(), ['name', 'rating']]
worst_cereal = df.loc[df['rating'].idxmin(), ['name', 'rating']]
print(f"Best cereal:\n{best_cereal}")
print(f"Worst cereal:\n{worst_cereal}")
```

```
Best cereal:
name      All-Bran_with_Extra_Fiber
rating                    93.704912
Name: 2, dtype: object
Worst cereal:
name      Cap'n'Crunch
rating       18.042851
Name: 9, dtype: object
```

In [6]:

```
# 5. Compare ratings for hot vs cold cereals
avg_rating_by_type = df.groupby('type')['rating'].mean()
print(f"Average rating for hot vs cold cereals:\n{avg_rating_by_type}")
```

```
Average rating for hot vs cold cereals:
type
C    41.734838
H    56.737708
Name: rating, dtype: float64
```

In [7]:

```
# 6. Cereals with highest fiber and lowest sugar
df_cleaned = df[df['sugars'] >= 0]
highest_fiber = df_cleaned.loc[df_cleaned['fiber'].idxmax(), ['name', 'fiber
lowest_sugar = df_cleaned[df_cleaned['sugars'] == df_cleaned['sugars'].min()
print(f"Cereal with highest fiber:\n{highest_fiber}")
print(f"\nCereal(s) with lowest sugar:\n{lowest_sugar}")
```

```
Cereal with highest fiber:
name     All-Bran_with_Extra_Fiber
fiber                         14.0
Name: 2, dtype: object

Cereal(s) with lowest sugar:
                         name  sugars
2   All-Bran_with_Extra_Fiber     0.0
19      Cream_of_Wheat_(Quick)     0.0
53                Puffed_Rice     0.0
54               Puffed_Wheat     0.0
62             Shredded_Wheat     0.0
63     Shredded_Wheat_'n'Bran     0.0
64  Shredded_Wheat_spoon_size     0.0
```

In [8]:

```python
# 7. Cereals with more than 3 grams of protein
high_protein_cereals = df[df['protein'] > 3][['name', 'protein']]
print("\nCereals with more than 3 grams of protein:")
print(high_protein_cereals)
```

```
Cereals with more than 3 grams of protein:
                                      name  protein
1                                 All-Bran        4
2                All-Bran_with_Extra_Fiber        4
10                                Cheerios        6
40                                    Life        4
42                                   Maypo        4
43    Muesli_Raisins,_Dates,_&amp;_Almonds        4
44   Muesli_Raisins,_Peaches,_&amp;_Pecans        4
55                       Quaker_Oat_Squares        4
56                          Quaker_Oatmeal        5
66                               Special_K        6
```

In [9]:

```python
# 8. Tabulate cereals by display shelf
display_shelf_counts = df['shelf'].value_counts()
print("\nCereals by display shelf:")
print(display_shelf_counts)
```

```
Cereals by display shelf:
3    35
2    21
1    20
Name: shelf, dtype: int64
```

In [10]:

```python
# 9. Sugar content variation across brands
sugar_by_brand = df.groupby('mfr')['sugars'].mean()
print("\nAverage sugar content per manufacturer:")
print(sugar_by_brand)
```

```
Average sugar content per manufacturer:
mfr
A    3.000000
G    7.954545
K    7.565217
N    1.000000
P    8.777778
Q    6.142857
R    6.125000
Name: sugars, dtype: float64
```

In [11]:
```python
# 10. Average calories in cereals per manufacturer
avg_calories_per_mfr = df.groupby('mfr')['calories'].mean()
print("\nAverage calories per manufacturer:")
print(avg_calories_per_mfr)
```

```
Average calories per manufacturer:
mfr
A    100.000000
G    111.363636
K    108.695652
N     90.000000
P    108.888889
Q     95.000000
R    115.000000
Name: calories, dtype: float64
```

In [12]:
```python
# 11. Average nutritional content across all cereals
avg_nutritional_content = df[['calories', 'sugars', 'protein',
'fiber']].mean()
print("\nAverage nutritional content across all cereals:")
print(avg_nutritional_content)
```

```
Average nutritional content across all cereals:
calories    107.368421
sugars        7.040000
protein       2.526316
fiber         2.048684
dtype: float64
```

In [13]:
```python
# 12. Relationship between sugar and calories
sugar_calories_corr = df[['sugars', 'calories']].corr()
print("\nCorrelation between sugar and calories:")
print(sugar_calories_corr)
```

```
Correlation between sugar and calories:
            sugars  calories
sugars    1.000000  0.574758
calories  0.574758  1.000000
```

In [14]:
```python
# 13. Compare sugar content in high vs low rated cereals
median_rating = df['rating'].median()
high_rated_cereals = df[df['rating'] >= median_rating]['sugars'].mean()
low_rated_cereals = df[df['rating'] < median_rating]['sugars'].mean()
print("\nAverage sugar content in high vs low rated cereals:")
print("High rated cereals:", high_rated_cereals)
print("Low rated cereals:", low_rated_cereals)
```

```
Average sugar content in high vs low rated cereals:
High rated cereals: 3.7837837837837838
Low rated cereals: 10.210526315789474
```

In [15]:
```
# 14. Do healthy cereals have higher ratings?
healthy_cereals = df[(df['fiber'] > 3) & (df['sugars'] < 5)]
healthy_cereal_avg_rating = healthy_cereals['rating'].mean()
print("\nAverage rating for healthy cereals:",
healthy_cereal_avg_rating)
```

```
Average rating for healthy cereals: 84.0889305
```

In [16]:
```
# 15. Relationship between rating and display shelf
rating_by_shelf = df.groupby('shelf')['rating'].mean()
print("\nAverage rating by display shelf:")
print(rating_by_shelf)
```

```
Average rating by display shelf:
shelf
1    46.145439
2    34.972827
3    44.557662
Name: rating, dtype: float64
```