Cars_Dataset_Analysis

September 21, 2025

1 Cars Dataset Analysis

1.1 Awais Manzoor

1.2 Introduction

This analysis explores the Cars dataset to understand relationships between different features.We will use statistical summaries and visualizations to uncover patterns.
- **Dataset Information**

- **Entries (rows):** 428

- **Columns (features):** 15
- **Column Details**

#	Column	Non-Null Count	Dtype	Description
0	Make	428 non-null	object	Car manufacturer (e.g., Toyota, Ford)
1	Model	428 non-null	object	Car model name
2	Type	428 non-null	object	Vehicle type (e.g., Sedan, SUV, Truck)
3	Origin	428 non-null	object	Country/region of origin
4	DriveTrain	428 non-null	object	Type of drive (e.g., FWD, RWD, AWD)
5	MSRP	428 non-null	int64	Manufacturer's Suggested Retail Price
6	Invoice	428 non-null	int64	Dealer invoice price
7	EngineSize	428 non-null	float64	Engine size in liters
8	Cylinders	426 non-null	float64	Number of engine cylinders
9	Horsepower	428 non-null	int64	Engine power output
10	MPG_City	428 non-null	int64	Fuel efficiency (miles per gallon, city)

#	Column	Non-Null Count	Dtype	Description
11	MPG_Highway	428 non-null	int64	Fuel efficiency (miles per gallon, highway)
12	Weight	428 non-null	int64	Vehicle weight (in lbs)
13	Wheelbase	428 non-null	int64	Distance between front and rear axles (in inches)
14	Length	428 non-null	int64	Vehicle length (in inches)

1.2.1 Observations

- The dataset has **428 cars** described by **15 features**.
- Most columns are complete, except **Cylinders**, which has 2 missing values.
- Data types include **categorical (object)** and **numerical (int64, float64)**.
- It contains both **technical specs** (Horsepower, Engine Size, Weight) and **pricing info** (MSRP, Invoice).

1.3 *import libraries*

[1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

2 *Load the Dataset*

[13]:
```python
cars = pd.read_excel('c:/Users/hp/OneDrive - Higher Education Commission/
↪Desktop/Business Data Analytics/CARS.xlsx')
cars
```

[13]:
```
      Make                    Model   Type  Origin DriveTrain   MSRP  \
0    Acura                      MDX    SUV    Asia        All  36945
1    Acura           RSX Type S 2dr  Sedan    Asia      Front  23820
2    Acura                  TSX 4dr  Sedan    Asia      Front  26990
3    Acura                   TL 4dr  Sedan    Asia      Front  33195
4    Acura               3.5 RL 4dr  Sedan    Asia      Front  43755
..     …                        …      …       …          …      …
423  Volvo  C70 LPT convertible 2dr  Sedan  Europe      Front  40565
424  Volvo  C70 HPT convertible 2dr  Sedan  Europe      Front  42565
425  Volvo               S80 T6 4dr  Sedan  Europe      Front  45210
```

```
426  Volvo                        V40   Wagon  Europe      Front  26135
427  Volvo                       XC70   Wagon  Europe        All  35145

     Invoice  EngineSize  Cylinders  Horsepower  MPG_City  MPG_Highway  \
0      33337         3.5        6.0         265        17           23
1      21761         2.0        4.0         200        24           31
2      24647         2.4        4.0         200        22           29
3      30299         3.2        6.0         270        20           28
4      39014         3.5        6.0         225        18           24
..       …          …        …          …       …        …
423    38203         2.4        5.0         197        21           28
424    40083         2.3        5.0         242        20           26
425    42573         2.9        6.0         268        19           26
426    24641         1.9        4.0         170        22           29
427    33112         2.5        5.0         208        20           27

     Weight  Wheelbase  Length
0      4451        106     189
1      2778        101     172
2      3230        105     183
3      3575        108     186
4      3880        115     197
..      …         …      …
423    3450        105     186
424    3450        105     186
425    3653        110     190
426    2822        101     180
427    3823        109     186

[428 rows x 15 columns]
```

3 *Understand the Dataset*

[42]:
```
cars.head(3)
```

[42]:
```
    Make           Model   Type Origin DriveTrain   MSRP  Invoice  \
0  Acura             MDX    suv   asia        all  36945    33337
1  Acura  RSX Type S 2dr  sedan   asia      front  23820    21761
2  Acura         TSX 4dr  sedan   asia      front  26990    24647

   EngineSize  Cylinders  Horsepower  MPG_City  MPG_Highway  Weight  \
0         3.5        6.0         265        17           23    4451
1         2.0        4.0         200        24           31    2778
2         2.4        4.0         200        22           29    3230

   Wheelbase  Length
```

```
0          106      189
1          101      172
2          105      183
```

[6]:
```
cars.tail(3)
```

[6]:
```
      Make         Model    Type  Origin DriveTrain   MSRP  Invoice  EngineSize  \
425  Volvo    S80 T6 4dr   Sedan  Europe      Front  45210    42573         2.9
426  Volvo           V40   Wagon  Europe      Front  26135    24641         1.9
427  Volvo          XC70   Wagon  Europe        All  35145    33112         2.5

     Cylinders  Horsepower  MPG_City  MPG_Highway  Weight  Wheelbase  Length
425        6.0         268        19           26    3653        110     190
426        4.0         170        22           29    2822        101     180
427        5.0         208        20           27    3823        109     186
```

[7]:
```
cars.shape
```

[7]:
```
(428, 15)
```

[8]:
```
cars.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 428 entries, 0 to 427
Data columns (total 15 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Make         428 non-null    object
 1   Model        428 non-null    object
 2   Type         428 non-null    object
 3   Origin       428 non-null    object
 4   DriveTrain   428 non-null    object
 5   MSRP         428 non-null    int64
 6   Invoice      428 non-null    int64
 7   EngineSize   428 non-null    float64
 8   Cylinders    426 non-null    float64
 9   Horsepower   428 non-null    int64
 10  MPG_City     428 non-null    int64
 11  MPG_Highway  428 non-null    int64
 12  Weight       428 non-null    int64
 13  Wheelbase    428 non-null    int64
 14  Length       428 non-null    int64
dtypes: float64(2), int64(8), object(5)
memory usage: 50.3+ KB
```

[9]:
```
cars.describe()
```

[9]:
```
                MSRP        Invoice  EngineSize   Cylinders  Horsepower  \
count     428.000000     428.000000  428.000000  426.000000  428.000000
mean    32774.855140   30014.700935    3.196729    5.807512  215.885514
std     19431.716674   17642.117750    1.108595    1.558443   71.836032
min     10280.000000    9875.000000    1.300000    3.000000   73.000000
25%     20334.250000   18866.000000    2.375000    4.000000  165.000000
50%     27635.000000   25294.500000    3.000000    6.000000  210.000000
75%     39205.000000   35710.250000    3.900000    6.000000  255.000000
max    192465.000000  173560.000000    8.300000   12.000000  500.000000

         MPG_City  MPG_Highway       Weight   Wheelbase      Length
count  428.000000   428.000000   428.000000  428.000000  428.000000
mean    20.060748    26.843458  3577.953271  108.154206  186.362150
std      5.238218     5.741201   758.983215    8.311813   14.357991
min     10.000000    12.000000  1850.000000   89.000000  143.000000
25%     17.000000    24.000000  3104.000000  103.000000  178.000000
50%     19.000000    26.000000  3474.500000  107.000000  187.000000
75%     21.250000    29.000000  3977.750000  112.000000  194.000000
max     60.000000    66.000000  7190.000000  144.000000  238.000000
```

[3]:
```python
cars.describe().T
```

[3]:
```
             count          mean           std      min        25%      50%  \
MSRP         428.0  32774.855140  19431.716674  10280.0  20334.250  27635.0
Invoice      428.0  30014.700935  17642.117750   9875.0  18866.000  25294.5
EngineSize   428.0      3.196729      1.108595      1.3      2.375      3.0
Cylinders    426.0      5.807512      1.558443      3.0      4.000      6.0
Horsepower   428.0    215.885514     71.836032     73.0    165.000    210.0
MPG_City     428.0     20.060748      5.238218     10.0     17.000     19.0
MPG_Highway  428.0     26.843458      5.741201     12.0     24.000     26.0
Weight       428.0   3577.953271    758.983215   1850.0   3104.000   3474.5
Wheelbase    428.0    108.154206      8.311813     89.0    103.000    107.0
Length       428.0    186.362150     14.357991    143.0    178.000    187.0

                  75%       max
MSRP         39205.00  192465.0
Invoice      35710.25  173560.0
EngineSize       3.90       8.3
Cylinders        6.00      12.0
Horsepower     255.00     500.0
MPG_City        21.25      60.0
MPG_Highway     29.00      66.0
Weight        3977.75    7190.0
Wheelbase      112.00     144.0
Length         194.00     238.0
```

[4]:
```python
cars.columns
```

[4]: Index(['Make', 'Model', 'Type', 'Origin', 'DriveTrain', 'MSRP', 'Invoice',
       'EngineSize', 'Cylinders', 'Horsepower', 'MPG_City', 'MPG_Highway',
       'Weight', 'Wheelbase', 'Length'],
      dtype='object')

4 Handle Missing Values

[14]:
```
cars.isnull().sum()
```

[14]:
```
Make           0
Model          0
Type           0
Origin         0
DriveTrain     0
MSRP           0
Invoice        0
EngineSize     0
Cylinders      2
Horsepower     0
MPG_City       0
MPG_Highway    0
Weight         0
Wheelbase      0
Length         0
dtype: int64
```

[15]:
```
cars.isnull().sum().sum()
```

[15]: np.int64(2)

[16]:
```
# Fill missing values in Cylinders with median
cars['Cylinders'] = cars['Cylinders'].fillna(cars['Cylinders'].median())
```

[17]:
```
cars.isnull().sum()
```

[17]:
```
Make           0
Model          0
Type           0
Origin         0
DriveTrain     0
MSRP           0
Invoice        0
EngineSize     0
Cylinders      0
Horsepower     0
MPG_City       0
```

```
MPG_Highway    0
Weight         0
Wheelbase      0
Length         0
dtype: int64
```

5 Remove Duplicates

[18]:
```
cars.duplicated().sum()
```

[18]:
```
np.int64(0)
```

6 Standardize Categories

[20]:
```
# Convert categorical columns to lowercase
cars['Type'] = cars['Type'].str.lower()
cars['Origin'] = cars['Origin'].str.lower()
cars['DriveTrain'] = cars['DriveTrain'].str.lower()
```

[ ]:
```

```

7 *Selecting a specific columns*

[21]:
```
cars['MSRP']
```

[21]:
```
0      36945
1      23820
2      26990
3      33195
4      43755
       …
423    40565
424    42565
425    45210
426    26135
427    35145
Name: MSRP, Length: 428, dtype: int64
```

[22]:
```
cars.Make
```

[22]:
```
0      Acura
1      Acura
2      Acura
3      Acura
```

```
4       Acura
        …
423     Volvo
424     Volvo
425     Volvo
426     Volvo
427     Volvo
Name: Make, Length: 428, dtype: object
```

[23]:
```
cars['EngineSize'].mean()
```

[23]:
```
np.float64(3.1967289719626164)
```

[24]:
```
# other method
cars.EngineSize.mean()
```

[24]:
```
np.float64(3.1967289719626164)
```

8 For non_numeric

[16]:
```
cars.mean()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[16], line 1
----> 1 cars.mean()

File c:\Users\hp\anaconda3\Lib\site-packages\pandas\core\frame.py:11693, in
 ↪DataFrame.mean(self, axis, skipna, numeric_only, **kwargs)
  11685 @doc(make_doc("mean", ndim=2))
  11686 def mean(
  11687     self,
   (…)
  11691     **kwargs,
  11692 ):
> 11693     result = super().mean(axis, skipna, numeric_only, **kwargs)
  11694     if isinstance(result, Series):
  11695         result = result.__finalize__(self, method="mean")

File c:\Users\hp\anaconda3\Lib\site-packages\pandas\core\generic.py:12420, in
 ↪NDFrame.mean(self, axis, skipna, numeric_only, **kwargs)
  12413 def mean(
  12414     self,
  12415     axis: Axis | None = 0,
   (…)
  12418     **kwargs,
```

```
  12419 ) -> Series | float:
> 12420     return self._stat_function(
  12421         "mean", nanops.nanmean, axis, skipna, numeric_only, **kwargs
  12422     )

File c:\Users\hp\anaconda3\Lib\site-packages\pandas\core\generic.py:12377, in␣
↪NDFrame._stat_function(self, name, func, axis, skipna, numeric_only, **kwargs
  12373 nv.validate_func(name, (), kwargs)
  12375 validate_bool_kwarg(skipna, "skipna", none_allowed=False)
> 12377 return self._reduce(
  12378     func, name=name, axis=axis, skipna=skipna, numeric_only=numeric_onl
  12379 )

File c:\Users\hp\anaconda3\Lib\site-packages\pandas\core\frame.py:11562, in␣
↪DataFrame._reduce(self, op, name, axis, skipna, numeric_only, filter_type,␣
↪**kwds)
  11558     df = df.T
  11560 # After possibly _get_data and transposing, we are now in the
  11561 #  simple case where we can use BlockManager.reduce
> 11562 res = df._mgr.reduce(blk_func)
  11563 out = df._constructor_from_mgr(res, axes=res.axes).iloc[0]
  11564 if out_dtype is not None and out.dtype != "boolean":

File c:\Users\hp\anaconda3\Lib\site-packages\pandas\core\internals\managers.py:
↪1500, in BlockManager.reduce(self, func)
   1498 res_blocks: list[Block] = []
   1499 for blk in self.blocks:
-> 1500     nbs = blk.reduce(func)
   1501     res_blocks.extend(nbs)
   1503 index = Index([None])  # placeholder

File c:\Users\hp\anaconda3\Lib\site-packages\pandas\core\internals\blocks.py:
↪404, in Block.reduce(self, func)
    398 @final
    399 def reduce(self, func) -> list[Block]:
    400     # We will apply the function and reshape the result into a single-row
    401     #  Block with the same mgr_locs; squeezing will be done at a higher␣
↪level
    402     assert self.ndim == 2
--> 404     result = func(self.values)
    406     if self.values.ndim == 1:
    407         res_values = result

File c:\Users\hp\anaconda3\Lib\site-packages\pandas\core\frame.py:11481, in␣
↪DataFrame._reduce.<locals>.blk_func(values, axis)
  11479         return np.array([result])
  11480 else:
> 11481     return op(values, axis=axis, skipna=skipna, **kwds)
```

```
File c:\Users\hp\anaconda3\Lib\site-packages\pandas\core\nanops.py:147, in␣
 ↪bottleneck_switch.__call__.<locals>.f(values, axis, skipna, **kwds)
    145         result = alt(values, axis=axis, skipna=skipna, **kwds)
    146 else:
--> 147     result = alt(values, axis=axis, skipna=skipna, **kwds)
    149 return result

File c:\Users\hp\anaconda3\Lib\site-packages\pandas\core\nanops.py:404, in␣
 ↪_datetimelike_compat.<locals>.new_func(values, axis, skipna, mask, **kwargs)
    401 if datetimelike and mask is None:
    402     mask = isna(values)
--> 404 result = func(values, axis=axis, skipna=skipna, mask=mask, **kwargs)
    406 if datetimelike:
    407     result = _wrap_results(result, orig_values.dtype, fill_value=iNaT)

File c:\Users\hp\anaconda3\Lib\site-packages\pandas\core\nanops.py:720, in␣
 ↪nanmean(values, axis, skipna, mask)
    718 count = _get_counts(values.shape, mask, axis, dtype=dtype_count)
    719 the_sum = values.sum(axis, dtype=dtype_sum)
--> 720 the_sum = _ensure_numeric(the_sum)
    722 if axis is not None and getattr(the_sum, "ndim", False):
    723     count = cast(np.ndarray, count)

File c:\Users\hp\anaconda3\Lib\site-packages\pandas\core\nanops.py:1686, in␣
 ↪_ensure_numeric(x)
   1683 inferred = lib.infer_dtype(x)
   1684 if inferred in ["string", "mixed"]:
   1685     # GH#44008, GH#36703 avoid casting e.g. strings to numeric
-> 1686     raise TypeError(f"Could not convert {x} to numeric")
   1687 try:
   1688     x = x.astype(np.complex128)

TypeError: Could not convert␣
 ↪['AcuraAcuraAcuraAcuraAcuraAcuraAcuraAudiAudiAudiAudiAudiAudiAudiAudiAudi udiAudiAudiAu
 ↪RoverLand RoverLand␣
 ↪RoverLexusLexusLexusLexusLexusLexusLexusLexusLexusLexusLexusLincolnLincolnLin olnLincolnLi
```

```
' MDX RSX Type S 2dr TSX 4dr TL 4dr 3.5 RL 4dr 3.5 RL w/Navigation 4dr NSX␣
↪coupe 2dr manual S A4 1.8T 4dr A41.8T convertible 2dr A4 3.0 4dr A4 3.0␣
↪Quattro 4dr manual A4 3.0 Quattro 4dr auto A6 3.0 4dr A6 3.0 Quattro 4dr A4 3
↪0 convertible 2dr A4 3.0 Quattro convertible 2dr A6 2.7 Turbo Quattro 4dr A6
↪2 Quattro 4dr A8 L Quattro 4dr S4 Quattro 4dr RS 6 4dr TT 1.8 convertible 2dr
↪(coupe) TT 1.8 Quattro 2dr (convertible) TT 3.2 coupe 2dr (convertible) A6 3.
↪Avant Quattro S4 Avant Quattro X3 3.0i X5 4.4i 325i 4dr 325Ci 2dr 325Ci␣
↪convertible 2dr 325xi 4dr 330i 4dr 330Ci 2dr 330xi 4dr 525i 4dr 330Ci␣
↪convertible 2dr 530i 4dr 545iA 4dr 745i 4dr 745Li 4dr M3 coupe 2dr M3␣
↪convertible 2dr Z4 convertible 2.5i 2dr Z4 convertible 3.0i 2dr 325xi Sport␣
↪Rainier Rendezvous CX Century Custom 4dr LeSabre Custom 4dr Regal LS 4dr Rega
↪GS 4dr LeSabre Limited 4dr Park Avenue 4dr Park Avenue Ultra 4dr Escalade SRX
↪V8 CTS VVT 4dr Deville 4dr Deville DTS 4dr Seville SLS 4dr XLR convertible 2d
↪Escalade EXT Suburban 1500 LT Tahoe LT TrailBlazer LT Tracker Aveo 4dr Aveo L
↪4dr hatch Cavalier 2dr Cavalier 4dr Cavalier LS 2dr Impala 4dr Malibu 4dr␣
↪Malibu LS 4dr Monte Carlo LS 2dr Impala LS 4dr Impala SS 4dr Malibu LT 4dr␣
↪Monte Carlo SS 2dr Astro Venture LS Corvette 2dr Corvette convertible 2dr␣
↪Avalanche 1500 Colorado Z85 Silverado 1500 Regular Cab Silverado SS SSR Malib
↪Maxx LS PT Cruiser 4dr PT Cruiser Limited 4dr Sebring 4dr Sebring Touring 4dr
↪300M 4dr Concorde LX 4dr Concorde LXi 4dr PT Cruiser GT 4dr Sebring␣
↪convertible 2dr 300M Special Edition 4dr Sebring Limited convertible 2dr Town
↪and Country LX Town and Country Limited Crossfire 2dr Pacifica Durango SLT␣
↪Neon SE 4dr Neon SXT 4dr Intrepid SE 4dr Stratus SXT 4dr Stratus SE 4dr␣
↪Intrepid ES 4dr Caravan SE Grand Caravan SXT Viper SRT-10 convertible 2dr␣
↪Dakota Regular Cab Dakota Club Cab Ram 1500 Regular Cab ST Excursion 6.8 XLT␣
↪Expedition 4.6 XLT Explorer XLT V6 Escape XLS Focus ZX3 2dr hatch Focus LX 4d
↪Focus SE 4dr Focus ZX5 5dr Focus SVT 2dr Taurus LX 4dr Taurus SES Duratec 4dr
↪Crown Victoria 4dr Crown Victoria LX 4dr Crown Victoria LX Sport 4dr Freestar
↪SE Mustang 2dr (convertible) Mustang GT Premium convertible 2dr Thunderbird␣
↪Deluxe convert w/hardtop 2d F-150 Regular Cab XL F-150 Supercab Lariat Ranger
↪2.3 XL Regular Cab Focus ZTW Taurus SE Envoy XUV SLE Yukon 1500 SLE Yukon XL␣
↪2500 SLT Safari SLE Canyon Z85 SL Regular Cab Sierra Extended Cab 1500 Sierra
↪HD 2500 Sonoma Crew Cab Civic Hybrid 4dr manual (gas/electric) Insight 2dr␣
↪(gas/electric) Pilot LX CR-V LX Element LX Civic DX 2dr Civic HX 2dr Civic LX
↪4dr Accord LX 2dr Accord EX 2dr Civic EX 4dr Civic Si 2dr hatch Accord LX V6␣
↪4dr Accord EX V6 2dr Odyssey LX Odyssey EX S2000 convertible 2dr H2 Santa Fe␣
↪GLS Accent 2dr hatch Accent GL 4dr Accent GT 2dr hatch Elantra GLS 4dr Elantr
↪GT 4dr Elantra GT 4dr hatch Sonata GLS 4dr Sonata LX 4dr XG350 4dr XG350 L 4d
↪Tiburon GT V6 2dr G35 4dr G35 Sport Coupe 2dr G35 4dr I35 4dr M45 4dr Q45␣
↪Luxury 4dr FX35 FX45 Ascender S Rodeo S X-Type 2.5 4dr X-Type 3.0 4dr S-Type 3.
↪0 4dr S-Type 4.2 4dr S-Type R 4dr Vanden Plas 4dr XJ8 4dr XJR 4dr XK8 coupe␣
↪2dr XK8 convertible 2dr XKR coupe 2dr XKR convertible 2dr Grand Cherokee␣
↪Laredo Liberty Sport Wrangler Sahara convertible 2dr Sorento LX Optima LX 4dr
↪Rio 4dr manual Rio 4dr auto Spectra 4dr Spectra GS 4dr hatch Spectra GSX 4dr␣
↪hatch Optima LX V6 4dr Amanti 4dr Sedona LX Rio Cinco Range Rover HSE␣
↪Discovery SE Freelander SE GX 470 LX 470 RX 330 ES 330 4dr IS 300 4dr manual␣
↪IS 300 4dr auto GS 300 4dr GS 430 4dr LS 430 4dr SC 430 convertible 2dr IS 30
↪SportCross Navigator Luxury Aviator Ultimate LS V6 Luxury 4dr LS V6 Premium␣
↪4dr LS V8 Sport 4dr LS V8 Ultimate 4dr Town Car Signature 4dr Town Car␣
↪Ultimate 4dr Town Car Ultimate L 4dr Cooper Cooper S Tribute DX 2.0 Mazda3 i␣
↪4dr Mazda3 s 4dr Mazda6 i 4dr MPV ES MX-5 Miata convertible 2dr MX-5 Miata LS
↪convertible 2dr RX-8 4dr automatic RX-8 4dr manual B2300 SX Regular Cab B4000
↪SE Cab Plus G500 ML500 C230 Sport 2dr C320 Sport 2dr C240 4dr C240 4dr C320␣
↪Sport 4dr C320 4dr C320 4dr C32 AMG 4dr CL500 2dr CL600 2dr CLK320 coupe 2dr␣
↪(convertible) CLK500 coupe 2dr (convertible) E320 4dr E500 4dr S430 4dr S500␣
↪4dr SL500 convertible 2dr SL55 AMG 2dr SL600 convertible 2dr SLK230␣
↪convertible 2dr SLK32 AMG 2dr C240 E320 E500 Mountaineer Sable GS 4dr Grand␣
↪Marquis GS 4dr Grand Marquis LS Premium 4dr Sable LS Premium 4dr Grand Marqui
↪LS Ultimate 4dr Marauder 4dr Monterey Luxury Sable GS Endeavor XLS Montero XL
↪Outlander LS Lancer ES 4dr Lancer LS 4dr Galant ES 2.4L 4dr Lancer OZ Rally␣
↪4dr auto Diamante LS 4dr Galant GTS 4dr Eclipse GTS 2dr Eclipse Spyder GT␣
↪convertible 2dr Lancer Evolution 4dr Lancer Sportback LS Pathfinder Armada SE
↪Pathfinder SE Xterra XE V6 Sentra 1.8 4dr Sentra 1.8 S 4dr Altima S 4dr Sentr
↪SE-R 4dr Altima SE 4dr Maxima SE 4dr Maxima SL 4dr Quest S Quest SE 350Z coup
↪2dr 350Z Enthusiast convertible 2dr Frontier King Cab XE V6 Titan King Cab XE
↪Murano SL Alero GX 2dr Alero GLS 2dr Silhouette GL Aztekt Sunfire 1SA 2dr␣
↪Grand Am GT 2dr Grand Prix GT1 4dr Sunfire 1SC 2dr Grand Prix GT2 4dr␣
↪Bonneville GXP 4dr Montana Montana EWB GTO 2dr Vibe Cayenne S 911 Carrera␣
↪convertible 2dr (coupe) 911 Carrera 4S coupe 2dr (convert) 911 Targa coupe 2d
↪911 GT2 2dr Boxster convertible 2dr Boxster S convertible 2dr 9-3 Arc Sport␣
↪4dr 9-3 Aero 4dr 9-5 Arc 4dr 9-5 Aero 4dr 9-3 Arc convertible 2dr 9-3 Aero␣
↪convertible 2dr 9-5 Aero VUE Ion1 4dr lon2 4dr lon3 4dr lon2 quad coupe 2dr␣
↪lon3 quad coupe 2dr L300-2 4dr L300 2 xA 4dr hatch xB Impreza 2.5 RS 4dr␣
↪Legacy L 4dr Legacy GT 4dr Outback Limited Sedan 4dr Outback H6 4dr Outback␣
↪H-6 VDC 4dr Impreza WRX 4dr Impreza WRX STi 4dr Baja Forester X Outback XL-7␣
↪EX Vitara LX Aeno S 4dr Aerio LX 4dr Forenza S 4dr Forenza EX 4dr Verona LX␣
↪4dr Aerio SX Prius 4dr (gas/electric) Sequoia SR5 4Runner SR5 V6 Highlander V
↪Land Cruiser RAV4 Corolla CE 4dr Corolla S 4dr Corolla LE 4dr Echo 2dr manual
↪Echo 2dr auto Echo 4dr Camry LE 4dr Camry LE V6 4dr Camry Solara SE 2dr Camry
```

```
↪'SUVSedanSedanSedanSedanSedanSportsSedanSedanSedanSedanSedanSedanSedanSedanSe danSedanSedanS

↪'AsiaAsiaAsiaAsiaAsiaAsiaAsiaEuropeEuropeEuropeEuropeEuropeEuropeEuropeEurope EuropeEuropeEu

↪'AllFrontFrontFrontFrontFrontRearFrontFrontFrontAllAllFrontAllFrontAllAllAllA lAllFrontFro
↪to numeric
```

8.0.1 For a non-numeric dataset, pandas gives us an error

[25]:
```python
# For non_numeric we use
cars.mean(numeric_only=True)
```

[25]:
```
MSRP           32774.855140
Invoice        30014.700935
EngineSize         3.196729
Cylinders          5.808411
Horsepower       215.885514
MPG_City          20.060748
MPG_Highway       26.843458
Weight          3577.953271
Wheelbase        108.154206
Length           186.362150
dtype: float64
```

[ ]:
```python
# Axis = 0 means along the columns
# Axis = 1 means along the rows
cars.mean(axis=1,numeric_only=True)
```

[ ]:
```
0      7534.25
1      4889.30
2      5541.24
3      6769.02
4      8723.75
        …
423    8276.24
424    8668.43
425    9205.79
426    5410.59
427    7263.75
Length: 428, dtype: float64
```

8.1 origion of car

[ ]:
```python
cars.Origin.unique()
```

[ ]: array(['Asia', 'Europe', 'USA'], dtype=object)

[ ]:
```
cars.Origin.nunique()
```

[ ]: 3

[ ]:
```
# To find origion of car
cars.Origin.value_counts()
```

[ ]:
```
Origin
Asia      158
USA       147
Europe    123
Name: count, dtype: int64
```

9 *Groupby Analysis*

[ ]:
```
# average engine size of cars based on origin
cars.groupby('Origin').mean(['EngineSize'])
```

[ ]:
```
                MSRP       Invoice  EngineSize  Cylinders  Horsepower  \
Origin
Asia    24741.322785  22602.177215    2.774051   5.185897  190.702532
Europe  48349.796748  44395.081301    3.206504   6.235772  251.894309
USA     28377.442177  25949.340136    3.642857   6.108844  212.823129

         MPG_City  MPG_Highway       Weight   Wheelbase      Length
Origin
Asia    22.012658    28.265823  3319.316456  105.886076  182.816456
Europe  18.731707    26.008130  3680.723577  106.447154  181.845528
USA     19.074830    26.013605  3769.952381  112.020408  193.952381
```

[ ]:
```
# focus just one coulmn
cars.groupby('Origin')['EngineSize'].mean()
```

[ ]:
```
Origin
Asia      2.774051
Europe    3.206504
USA       3.642857
Name: EngineSize, dtype: float64
```

10 *Median*

[ ]:
```
cars['EngineSize'].median()
```

[ ]: 3.0

[ ]:
```
print(cars.median(numeric_only=True))
```

```
MSRP          27635.0
Invoice       25294.5
EngineSize        3.0
Cylinders         6.0
Horsepower      210.0
MPG_City         19.0
MPG_Highway      26.0
Weight         3474.5
Wheelbase       107.0
Length          187.0
dtype: float64
```

[ ]:
```
print(cars.median(axis=1,numeric_only=True))
```

```
0      147.5
1      136.5
2      144.0
3      147.0
4      156.0
       …
423    145.5
424    145.5
425    150.0
426    135.5
427    147.5
Length: 428, dtype: float64
```

11 *Mode*

[ ]:
```
cars.mode(axis=0,numeric_only=False,dropna=True)
```

[ ]:

	Make	Model	Type	Origin	DriveTrain	MSRP	Invoice	EngineSize	\
0	Toyota	C240 4dr	Sedan	Asia	Front	13270	14207.0	3.0	
1	NaN	C320 4dr	NaN	NaN	NaN	15389	19638.0	NaN	
2	NaN	G35 4dr	NaN	NaN	NaN	19635	68306.0	NaN	
3	NaN	NaN	NaN	NaN	NaN	19860	NaN	NaN	
4	NaN	NaN	NaN	NaN	NaN	21055	NaN	NaN	
5	NaN	NaN	NaN	NaN	NaN	21595	NaN	NaN	
6	NaN	NaN	NaN	NaN	NaN	23495	NaN	NaN	
7	NaN	NaN	NaN	NaN	NaN	23895	NaN	NaN	
8	NaN	NaN	NaN	NaN	NaN	25700	NaN	NaN	
9	NaN	NaN	NaN	NaN	NaN	27490	NaN	NaN	
10	NaN	NaN	NaN	NaN	NaN	28495	NaN	NaN	

```
11     NaN          NaN     NaN     NaN          NaN   29995        NaN            NaN
12     NaN          NaN     NaN     NaN          NaN   31545        NaN            NaN
13     NaN          NaN     NaN     NaN          NaN   33995        NaN            NaN
14     NaN          NaN     NaN     NaN          NaN   34495        NaN            NaN
15     NaN          NaN     NaN     NaN          NaN   35940        NaN            NaN
16     NaN          NaN     NaN     NaN          NaN   49995        NaN            NaN
17     NaN          NaN     NaN     NaN          NaN   74995        NaN            NaN

    Cylinders  Horsepower  MPG_City  MPG_Highway  Weight  Wheelbase  Length
0         6.0       200.0      18.0         26.0  3175.0      107.0   178.0
1         NaN         NaN       NaN          NaN  3285.0        NaN     NaN
2         NaN         NaN       NaN          NaN  3450.0        NaN     NaN
3         NaN         NaN       NaN          NaN     NaN        NaN     NaN
4         NaN         NaN       NaN          NaN     NaN        NaN     NaN
5         NaN         NaN       NaN          NaN     NaN        NaN     NaN
6         NaN         NaN       NaN          NaN     NaN        NaN     NaN
7         NaN         NaN       NaN          NaN     NaN        NaN     NaN
8         NaN         NaN       NaN          NaN     NaN        NaN     NaN
9         NaN         NaN       NaN          NaN     NaN        NaN     NaN
10        NaN         NaN       NaN          NaN     NaN        NaN     NaN
11        NaN         NaN       NaN          NaN     NaN        NaN     NaN
12        NaN         NaN       NaN          NaN     NaN        NaN     NaN
13        NaN         NaN       NaN          NaN     NaN        NaN     NaN
14        NaN         NaN       NaN          NaN     NaN        NaN     NaN
15        NaN         NaN       NaN          NaN     NaN        NaN     NaN
16        NaN         NaN       NaN          NaN     NaN        NaN     NaN
17        NaN         NaN       NaN          NaN     NaN        NaN     NaN
```

[ ]:
```python
cars.mode(axis='columns',numeric_only=False,dropna=True)
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_1376\1809561256.py:1: UserWarning:
Unable to sort modes: '<' not supported between instances of 'int' and 'str'
  cars.mode(axis='columns',numeric_only=False,dropna=True)
```

[ ]:
```
          0                          1      2       3      4        5        6  \
0    Acura                         MDX    SUV    Asia    All  36945.0  33337.0
1    Acura                RSX Type S 2dr  Sedan    Asia  Front  23820.0  21761.0
2    Acura                     TSX 4dr  Sedan    Asia  Front  26990.0  24647.0
3    Acura                      TL 4dr  Sedan    Asia  Front  33195.0  30299.0
4    Acura                  3.5 RL 4dr  Sedan    Asia  Front  43755.0  39014.0
..     …                            …      …       …      …        …
423  Volvo   C70 LPT convertible 2dr  Sedan  Europe  Front  40565.0  38203.0
424  Volvo   C70 HPT convertible 2dr  Sedan  Europe  Front  42565.0  40083.0
425  Volvo                 S80 T6 4dr  Sedan  Europe  Front  45210.0  42573.0
426  Volvo                        V40  Wagon  Europe  Front  26135.0  24641.0
427  Volvo                       XC70  Wagon  Europe    All  35145.0  33112.0
```

```
       7    8      9     10    11      12     13     14
0    3.5  6.0  265.0  17.0  23.0  4451.0  106.0  189.0
1    2.0  4.0  200.0  24.0  31.0  2778.0  101.0  172.0
2    2.4  4.0  200.0  22.0  29.0  3230.0  105.0  183.0
3    3.2  6.0  270.0  20.0  28.0  3575.0  108.0  186.0
4    3.5  6.0  225.0  18.0  24.0  3880.0  115.0  197.0
..   …   …     …     …     …       …      …      …
423  2.4  5.0  197.0  21.0  28.0  3450.0  105.0  186.0
424  2.3  5.0  242.0  20.0  26.0  3450.0  105.0  186.0
425  2.9  6.0  268.0  19.0  26.0  3653.0  110.0  190.0
426  1.9  4.0  170.0  22.0  29.0  2822.0  101.0  180.0
427  2.5  5.0  208.0  20.0  27.0  3823.0  109.0  186.0

[428 rows x 15 columns]
```

[ ]:
```
mode = cars['EngineSize'].mode()
mode
```

[ ]:
```
0    3.0
Name: EngineSize, dtype: float64
```

[ ]:
```
mode = cars['EngineSize'].mode()[0]
mode
```

[ ]:
```
np.float64(3.0)
```

[ ]:
```
cars.min(axis=0,numeric_only=True)
```

[ ]:
```
MSRP           10280.0
Invoice         9875.0
EngineSize         1.3
Cylinders          3.0
Horsepower        73.0
MPG_City          10.0
MPG_Highway       12.0
Weight          1850.0
Wheelbase         89.0
Length           143.0
dtype: float64
```

[ ]:
```
cars.max(axis=0,numeric_only=True)
```

[ ]:
```
MSRP           192465.0
Invoice        173560.0
EngineSize          8.3
Cylinders          12.0
Horsepower        500.0
```

```
MPG_City           60.0
MPG_Highway        66.0
Weight           7190.0
Wheelbase         144.0
Length            238.0
dtype: float64
```

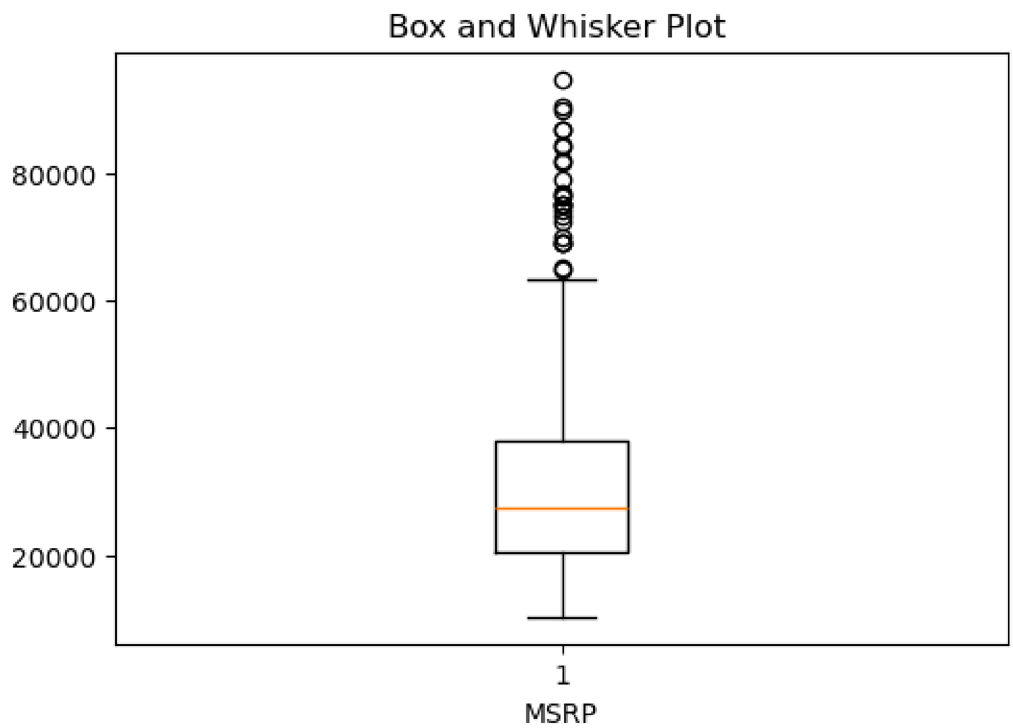
12 MSRP

[ ]:
```
cars['MSRP'].describe()
```

[ ]:
```
count       428.000000
mean      32774.855140
std       19431.716674
min       10280.000000
25%       20334.250000
50%       27635.000000
75%       39205.000000
max      192465.000000
Name: MSRP, dtype: float64
```

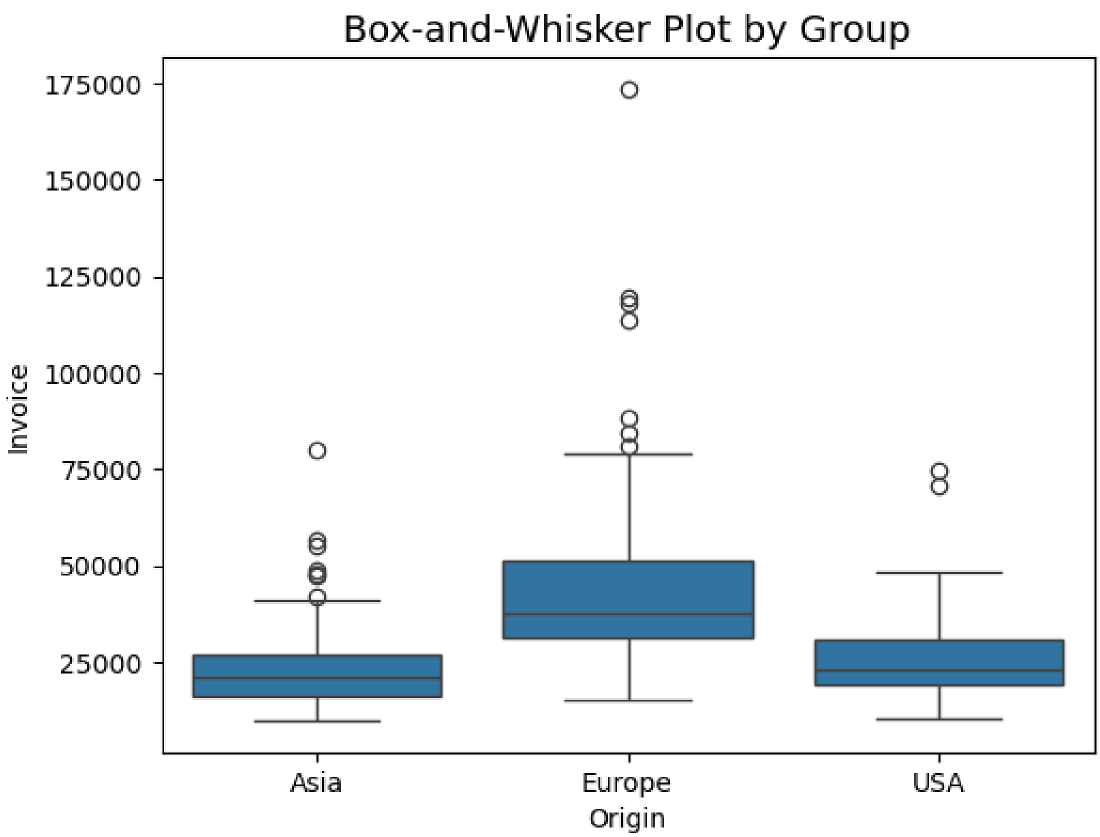
13 Handle Outliers

[29]:
```
plt.figure(figsize=(6, 4))
plt.boxplot(cars['MSRP'])
plt.title('Box and Whisker Plot ')
plt.xlabel('MSRP')
plt.show()
```



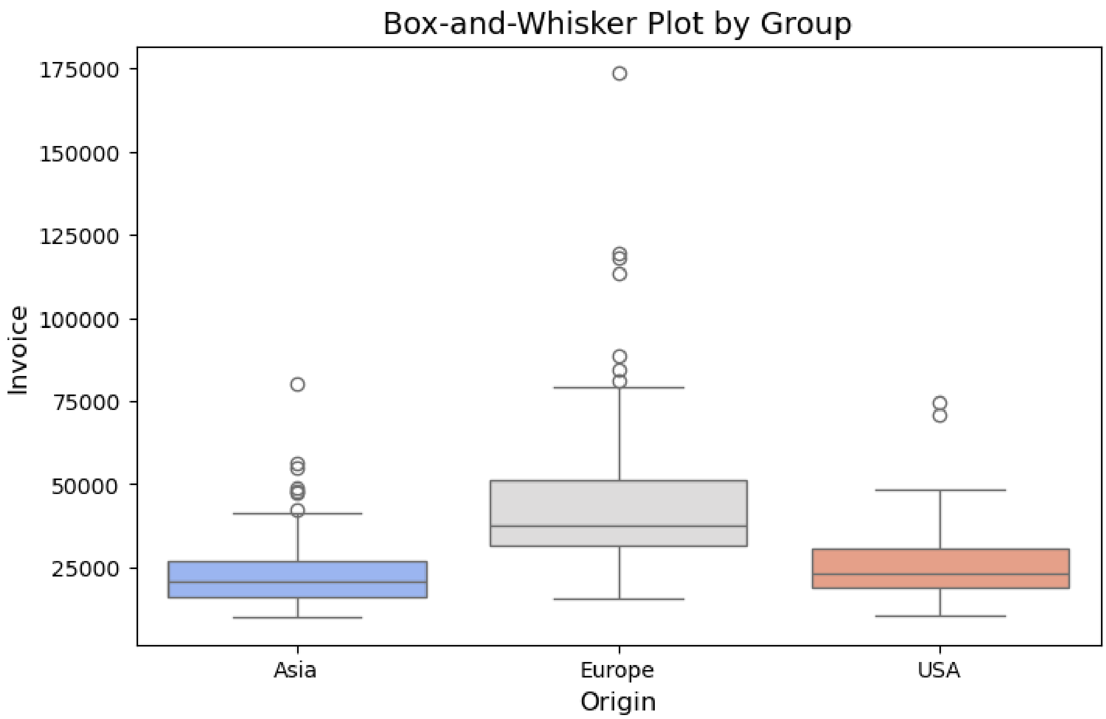

```
[ ]:
```

```
[ ]: sns.boxplot(x='Origin', y='Invoice', data=cars)
     plt.title('Box-and-Whisker Plot by Group', fontsize=14)
     plt.show()
```



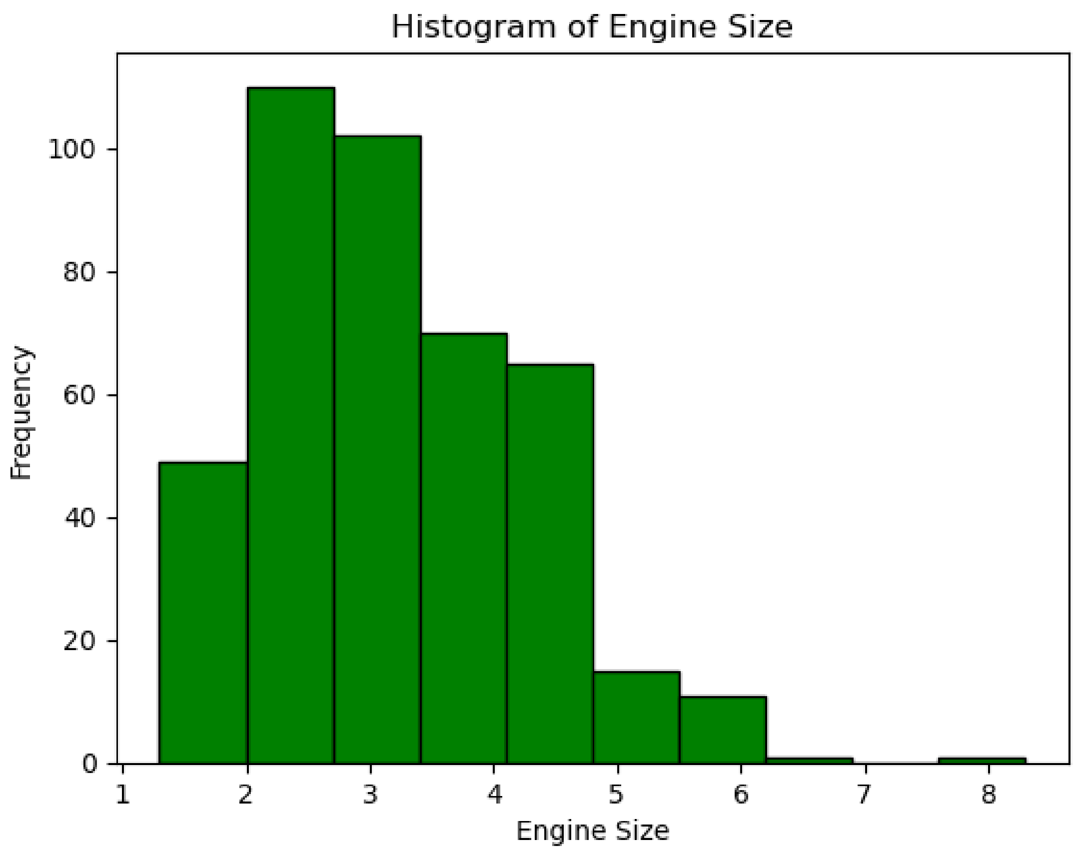
[ ]:
```
plt.figure(figsize=(8, 5))
sns.boxplot(x='Origin', y='Invoice', data=cars, palette='coolwarm')
plt.title('Box-and-Whisker Plot by Group', fontsize=14)
plt.xlabel('Origin', fontsize=12)
plt.ylabel('Invoice', fontsize=12)
plt.show()
```

```
C:\Users\hp\AppData\Local\Temp\ipykernel_1376\1166509147.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same
effect.

  sns.boxplot(x='Origin', y='Invoice', data=cars, palette='coolwarm')
```

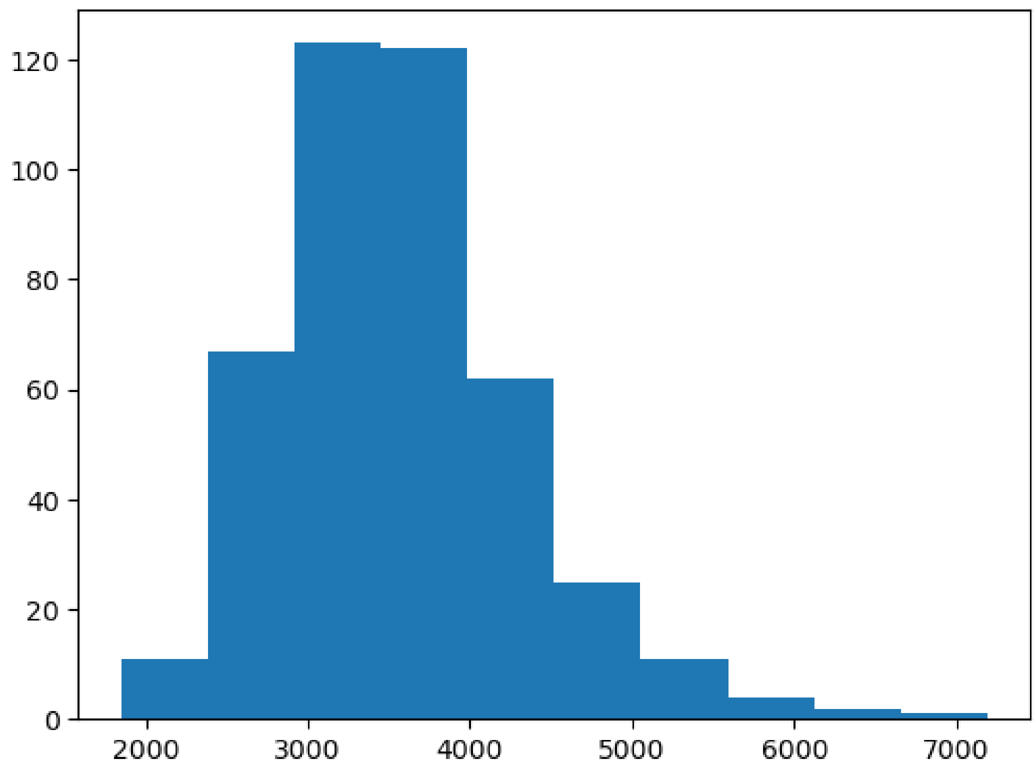


14 Histogram

[31]:
```
plt.hist(cars['EngineSize'], bins=10, color='green', edgecolor='black')
plt.title('Histogram of Engine Size')
plt.xlabel('Engine Size')
plt.ylabel('Frequency')
plt.show()
```

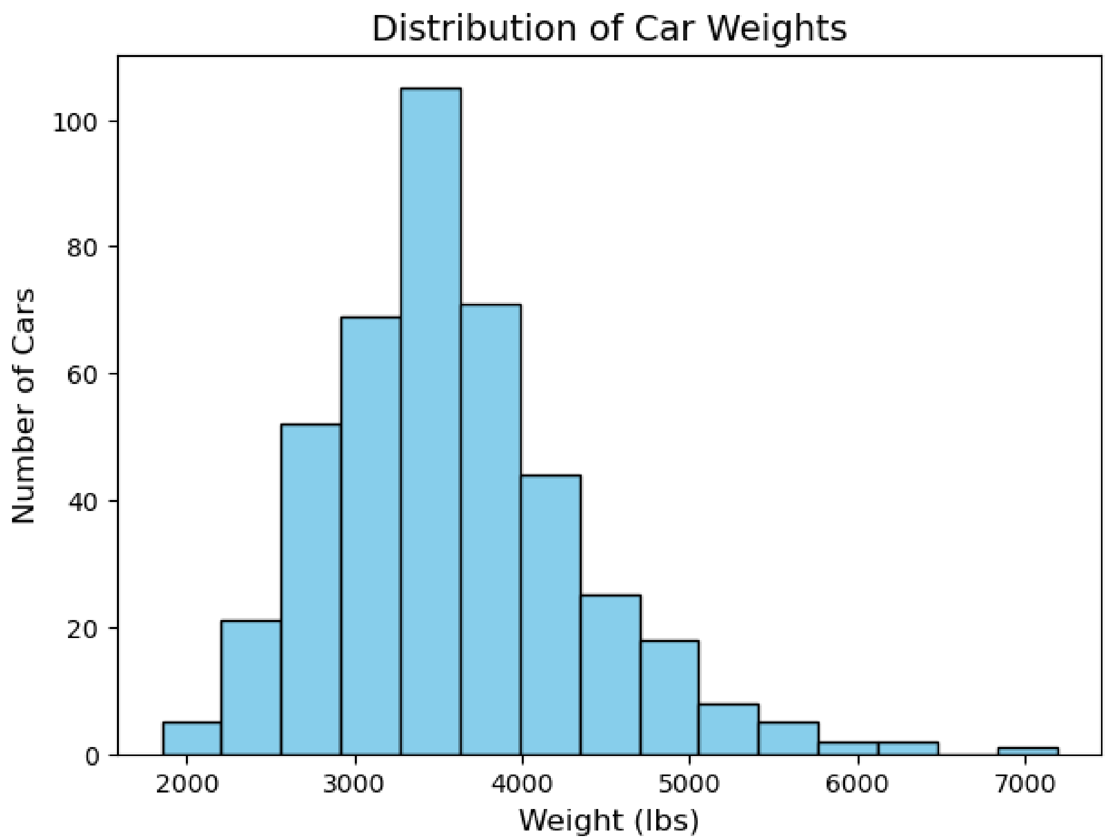


[ ]:
```
plt.hist(cars['Weight'])
plt.show()
```



[ ]:

```
plt.figure(figsize=(7,5))
plt.hist(cars['Weight'], bins=15, color='skyblue', edgecolor='black')
plt.title("Distribution of Car Weights", fontsize=14)
plt.xlabel("Weight (lbs)", fontsize=12)
plt.ylabel("Number of Cars", fontsize=12)
plt.show()
```



[35]:
```
import scipy.stats as stats
```

[36]:
```
numeric_data = cars.select_dtypes(include=['number'])
numeric_data
```

[36]:
```
      MSRP  Invoice  EngineSize  Cylinders  Horsepower  MPG_City  MPG_Highway  \
0    36945    33337         3.5        6.0         265        17           23
1    23820    21761         2.0        4.0         200        24           31
2    26990    24647         2.4        4.0         200        22           29
3    33195    30299         3.2        6.0         270        20           28
4    43755    39014         3.5        6.0         225        18           24
..     …        …          …          …          …         …            …
423  40565    38203         2.4        5.0         197        21           28
424  42565    40083         2.3        5.0         242        20           26
425  45210    42573         2.9        6.0         268        19           26
426  26135    24641         1.9        4.0         170        22           29
427  35145    33112         2.5        5.0         208        20           27

     Weight  Wheelbase  Length
0      4451        106     189
```

```
1      2778         101      172
2      3230         105      183
3      3575         108      186
4      3880         115      197
..      …          …      …
423    3450         105      186
424    3450         105      186
425    3653         110      190
426    2822         101      180
427    3823         109      186

[424 rows x 10 columns]
```

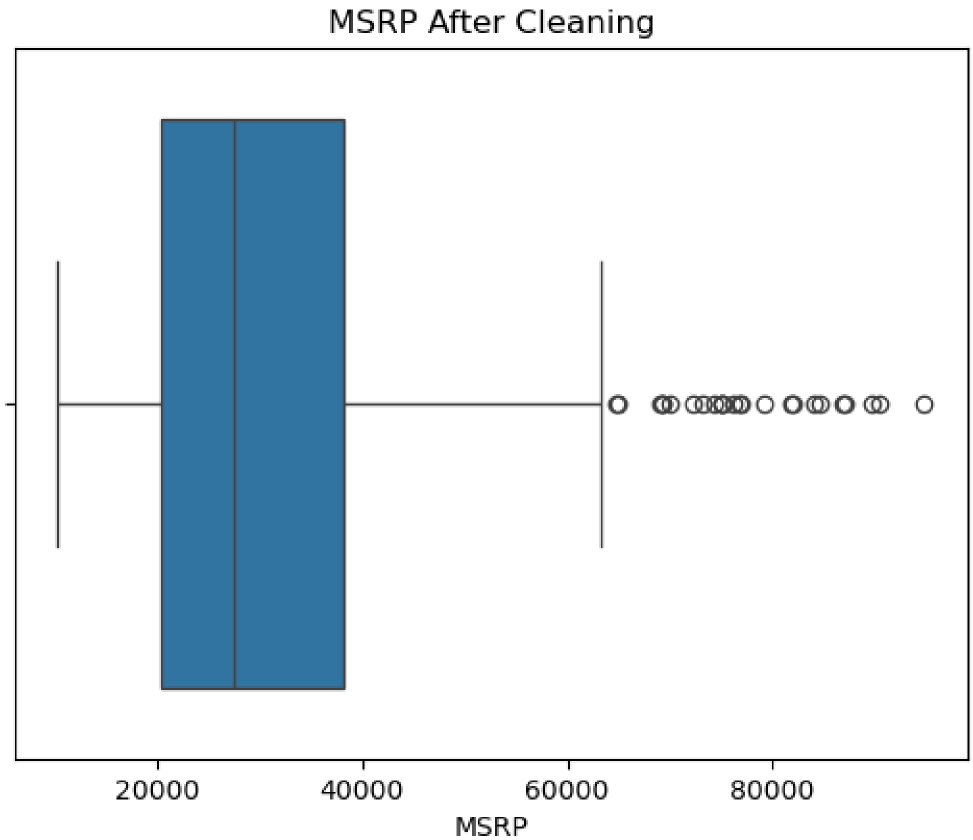
15 Feature Engineering

[43]:
```
# Add price difference
cars['Price_Difference'] = cars['MSRP'] - cars['Invoice']
```

[44]:
```
# Fuel efficiency ratio
cars['Efficiency_Ratio'] = cars['MPG_Highway'] / cars['MPG_City']
```

[45]:
```
# Horsepower category
cars['HP_Category'] = pd.cut(cars['Horsepower'],
                             bins=[0,150,300,500],
                             labels=['Low','Medium','High'])
```

[47]:
```
sns.boxplot(x=cars['MSRP'])
plt.title("MSRP After Cleaning")
plt.show()
```



16 Data Visualization

17 Z_Score

[37]:
```
z_df = numeric_data.apply(stats.zscore)
z_df
```

[37]:
```
         MSRP   Invoice  EngineSize  Cylinders  Horsepower  MPG_City  \
0    0.323966  0.294773    0.292658   0.151240    0.771754 -0.596438
1   -0.493156 -0.503249   -1.076956  -1.184712   -0.198704  0.743629
2   -0.295801 -0.304295   -0.711726  -1.184712   -0.198704  0.360753
3    0.090503  0.085341    0.018735   0.151240    0.846404 -0.022124
4    0.747936  0.686132    0.292658   0.151240    0.174549 -0.405000
..        …         …           …          …        …         …
423  0.549336  0.630224   -0.711726  -0.516736   -0.243495  0.169314
424  0.673850  0.759827   -0.803033  -0.516736    0.428361 -0.022124
425  0.838519  0.931481   -0.255188   0.151240    0.816544 -0.213562
426 -0.349031 -0.304709   -1.168264  -1.184712   -0.646608  0.360753
427  0.211904  0.279262   -0.620418  -0.516736   -0.079263 -0.022124
```

```
     MPG_Highway    Weight  Wheelbase    Length
0      -0.681119  1.157401  -0.266606  0.180721
1       0.715710 -1.048842  -0.869811 -1.001346
2       0.366503 -0.452774  -0.387247 -0.236479
3       0.191899  0.002190  -0.025323 -0.027879
4      -0.506515  0.404404   0.819165  0.736988
..           …         …          …        …
423     0.191899 -0.162652  -0.387247 -0.027879
424    -0.157308 -0.162652  -0.387247 -0.027879
425    -0.157308  0.105051   0.215959  0.250255
426     0.366503 -0.990817  -0.869811 -0.445079
427     0.017296  0.329236   0.095318 -0.027879

[424 rows x 10 columns]
```

[22]:

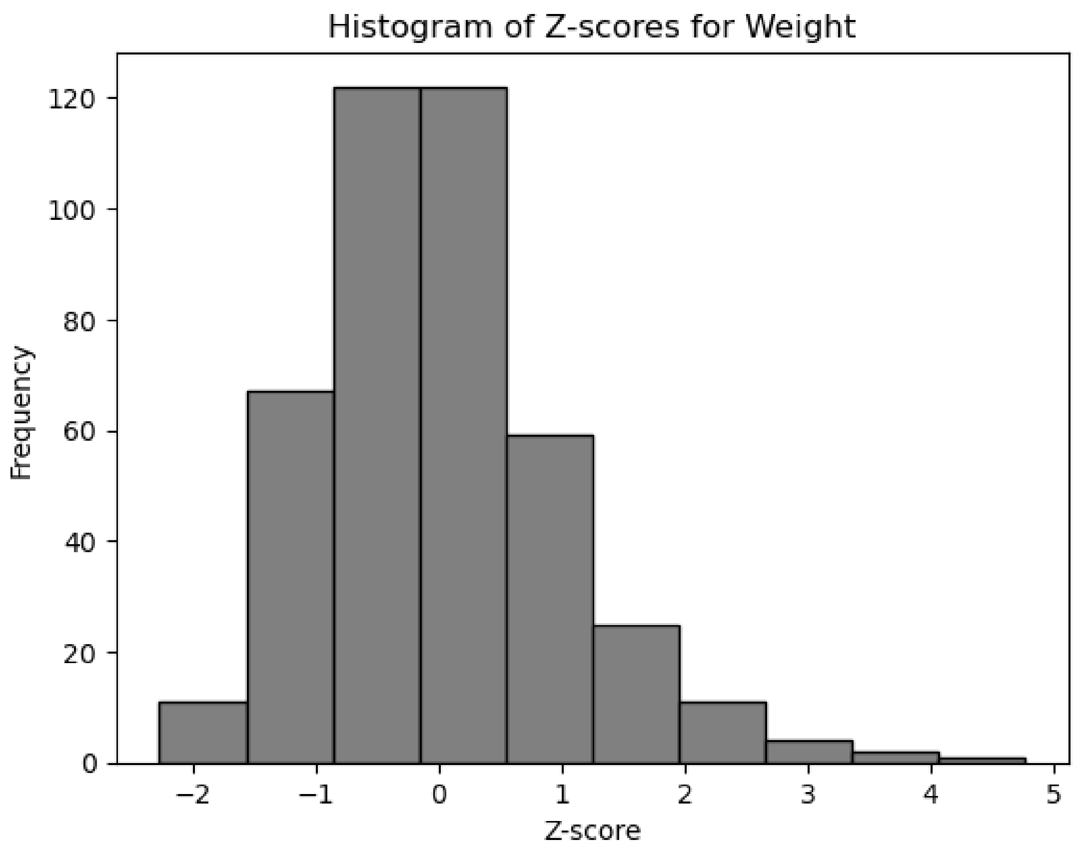
```
z_df .describe().round(2).T
```

[22]:

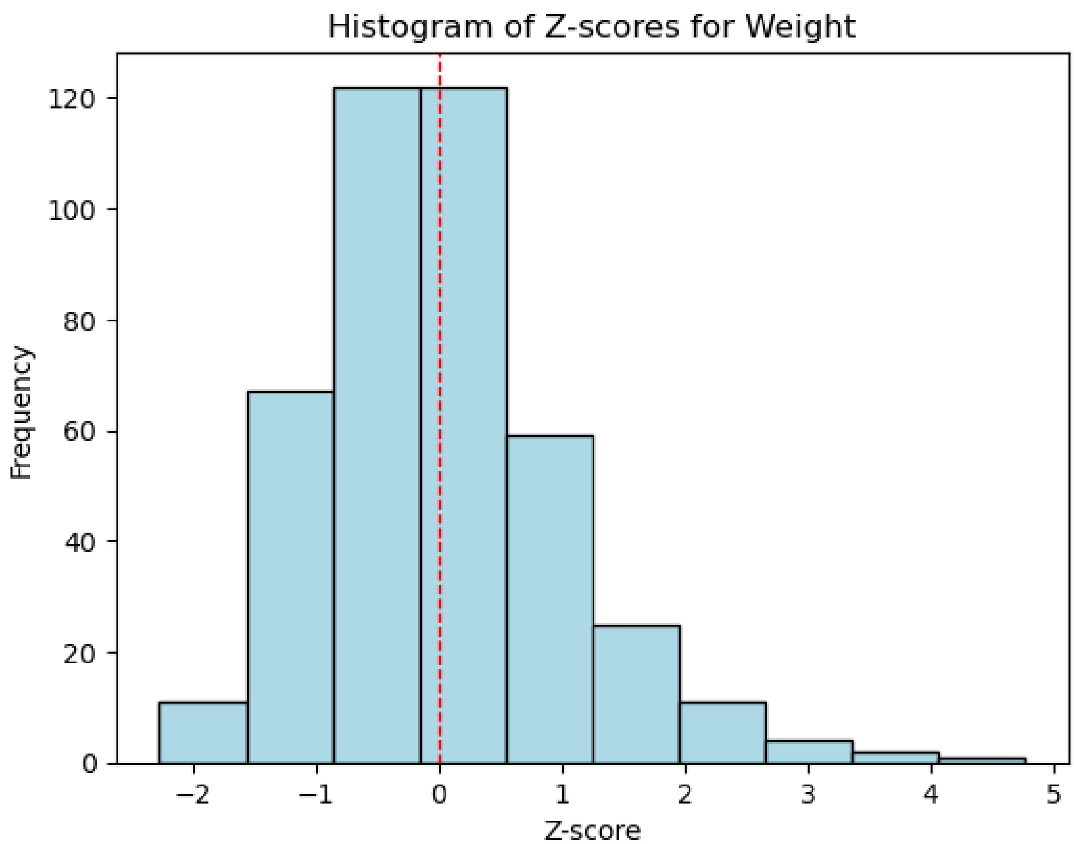
	count	mean	std	min	25%	50%	75%	max
MSRP	428.0	-0.0	1.0	-1.16	-0.64	-0.26	0.33	8.23
Invoice	428.0	0.0	1.0	-1.14	-0.63	-0.27	0.32	8.15
EngineSize	428.0	0.0	1.0	-1.71	-0.74	-0.18	0.64	4.61
Cylinders	0.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN
Horsepower	428.0	-0.0	1.0	-1.99	-0.71	-0.08	0.55	3.96
MPG_City	428.0	-0.0	1.0	-1.92	-0.58	-0.20	0.23	7.63
MPG_Highway	428.0	-0.0	1.0	-2.59	-0.50	-0.15	0.38	6.83
Weight	428.0	0.0	1.0	-2.28	-0.63	-0.14	0.53	4.76
Wheelbase	428.0	-0.0	1.0	-2.31	-0.62	-0.14	0.46	4.32
Length	428.0	0.0	1.0	-3.02	-0.58	0.04	0.53	3.60

[40]:

```
plt.hist(z_df['Weight'], bins=10, color='gray', edgecolor='black')
plt.title('Histogram of Z-scores for Weight')
plt.xlabel('Z-score')
plt.ylabel('Frequency')
plt.show()
```



[41]:
```
plt.hist(z_df['Weight'], bins=10, color='lightblue', edgecolor='black')
plt.title('Histogram of Z-scores for Weight')
plt.xlabel('Z-score')
plt.ylabel('Frequency')
plt.axvline(0, color='red', linestyle='dashed', linewidth=1)
plt.show()
```



[29]:
```
from scipy.stats import skew, kurtosis
```

[32]:
```
result = cars[cars['Invoice'] > 17000]
result
```

[32]:
```
      Make                        Model    Type  Origin DriveTrain   MSRP  \
0    Acura                          MDX     SUV    Asia        All  36945
1    Acura               RSX Type S 2dr   Sedan    Asia      Front  23820
2    Acura                      TSX 4dr   Sedan    Asia      Front  26990
3    Acura                       TL 4dr   Sedan    Asia      Front  33195
4    Acura                   3.5 RL 4dr   Sedan    Asia      Front  43755
..     …                            …     …     …        …      …
423  Volvo   C70 LPT convertible 2dr   Sedan  Europe      Front  40565
424  Volvo   C70 HPT convertible 2dr   Sedan  Europe      Front  42565
425  Volvo                   S80 T6 4dr   Sedan  Europe      Front  45210
426  Volvo                          V40   Wagon  Europe      Front  26135
427  Volvo                         XC70   Wagon  Europe        All  35145

     Invoice  EngineSize  Cylinders  Horsepower  MPG_City  MPG_Highway  \
0      33337         3.5        6.0         265        17           23
```

```
1      21761       2.0       4.0        200        24          31
2      24647       2.4       4.0        200        22          29
3      30299       3.2       6.0        270        20          28
4      39014       3.5       6.0        225        18          24
..       …       …        …        …        …        …
423    38203       2.4       5.0        197        21          28
424    40083       2.3       5.0        242        20          26
425    42573       2.9       6.0        268        19          26
426    24641       1.9       4.0        170        22          29
427    33112       2.5       5.0        208        20          27

     Weight  Wheelbase  Length
0      4451        106     189
1      2778        101     172
2      3230        105     183
3      3575        108     186
4      3880        115     197
..      …        …     …
423    3450        105     186
424    3450        105     186
425    3653        110     190
426    2822        101     180
427    3823        109     186

[357 rows x 15 columns]
```

[33]:
```python
cars['Invoice'].isnull().sum()
```

[33]:
```
np.int64(0)
```

[35]:
```python
cars.describe(include=[np.number]).T
```

[35]:
```
             count          mean           std      min        25%      50%  \
MSRP         428.0  32774.855140  19431.716674  10280.0  20334.250  27635.0
Invoice      428.0  30014.700935  17642.117750   9875.0  18866.000  25294.5
EngineSize   428.0      3.196729      1.108595      1.3      2.375      3.0
Cylinders    426.0      5.807512      1.558443      3.0      4.000      6.0
Horsepower   428.0    215.885514     71.836032     73.0    165.000    210.0
MPG_City     428.0     20.060748      5.238218     10.0     17.000     19.0
MPG_Highway  428.0     26.843458      5.741201     12.0     24.000     26.0
Weight       428.0   3577.953271    758.983215   1850.0   3104.000   3474.5
Wheelbase    428.0    108.154206      8.311813     89.0    103.000    107.0
Length       428.0    186.362150     14.357991    143.0    178.000    187.0

                  75%       max
MSRP         39205.00  192465.0
Invoice      35710.25  173560.0
```

```
EngineSize        3.90      8.3
Cylinders         6.00     12.0
Horsepower      255.00    500.0
MPG_City         21.25     60.0
MPG_Highway      29.00     66.0
Weight         3977.75   7190.0
Wheelbase       112.00    144.0
Length          194.00    238.0
```

[36]:
```python
cars.describe(include='all')
```

[36]:
```
          Make      Model   Type Origin DriveTrain           MSRP  \
count      428        428    428    428        428     428.000000
unique      38        425      6      3          3            NaN
top     Toyota  G35 4dr  Sedan   Asia      Front            NaN
freq        28          2    262    158        226            NaN
mean       NaN        NaN    NaN    NaN        NaN   32774.855140
std        NaN        NaN    NaN    NaN        NaN   19431.716674
min        NaN        NaN    NaN    NaN        NaN   10280.000000
25%        NaN        NaN    NaN    NaN        NaN   20334.250000
50%        NaN        NaN    NaN    NaN        NaN   27635.000000
75%        NaN        NaN    NaN    NaN        NaN   39205.000000
max        NaN        NaN    NaN    NaN        NaN  192465.000000

              Invoice  EngineSize   Cylinders  Horsepower    MPG_City  \
count      428.000000  428.000000  426.000000  428.000000  428.000000
unique            NaN         NaN         NaN         NaN         NaN
top               NaN         NaN         NaN         NaN         NaN
freq              NaN         NaN         NaN         NaN         NaN
mean     30014.700935    3.196729    5.807512  215.885514   20.060748
std      17642.117750    1.108595    1.558443   71.836032    5.238218
min       9875.000000    1.300000    3.000000   73.000000   10.000000
25%      18866.000000    2.375000    4.000000  165.000000   17.000000
50%      25294.500000    3.000000    6.000000  210.000000   19.000000
75%      35710.250000    3.900000    6.000000  255.000000   21.250000
max     173560.000000    8.300000   12.000000  500.000000   60.000000

        MPG_Highway       Weight   Wheelbase      Length
count    428.000000   428.000000  428.000000  428.000000
unique          NaN          NaN         NaN         NaN
top             NaN          NaN         NaN         NaN
freq            NaN          NaN         NaN         NaN
mean      26.843458  3577.953271  108.154206  186.362150
std        5.741201   758.983215    8.311813   14.357991
min       12.000000  1850.000000   89.000000  143.000000
25%       24.000000  3104.000000  103.000000  178.000000
50%       26.000000  3474.500000  107.000000  187.000000
```

```
75%        29.000000  3977.750000  112.000000  194.000000
max        66.000000  7190.000000  144.000000  238.000000
```

[37]:
```
# only non-numeric columns
cars.describe(exclude=[np.number])
```

[37]:
```
          Make     Model   Type Origin DriveTrain
count      428       428    428    428        428
unique      38       425      6      3          3
top     Toyota   G35 4dr  Sedan   Asia      Front
freq        28         2    262    158        226
```

[40]:
```
cars.columns
```

[40]:
```
Index(['Make', 'Model', 'Type', 'Origin', 'DriveTrain', 'MSRP', 'Invoice',
       'EngineSize', 'Cylinders', 'Horsepower', 'MPG_City', 'MPG_Highway',
       'Weight', 'Wheelbase', 'Length'],
      dtype='object')
```

[42]:
```
cars.describe(include=[np.number], percentiles=[.01, .05, .10, .25, .5, .75, .
↪90, .95, .99]).T
```

[42]:
```
             count          mean           std      min        1%        5%  \
MSRP         428.0  32774.855140  19431.716674  10280.0  11191.45  13691.00
Invoice      428.0  30014.700935  17642.117750   9875.0  10659.01  12836.65
EngineSize   428.0      3.196729      1.108595      1.3      1.50      1.70
Cylinders    426.0      5.807512      1.558443      3.0      4.00      4.00
Horsepower   428.0    215.885514     71.836032     73.0    103.00    115.00
MPG_City     428.0     20.060748      5.238218     10.0     12.00     14.00
MPG_Highway  428.0     26.843458      5.741201     12.0     16.27     18.00
Weight       428.0   3577.953271    758.983215   1850.0   2211.20   2513.00
Wheelbase    428.0    108.154206      8.311813     89.0     93.00     95.35
Length       428.0    186.362150     14.357991    143.0    153.27    163.00

                 10%        25%      50%       75%      90%       95%  \
MSRP         15484.5  20334.250  27635.0  39205.00  52781.0  72864.25
Invoice      14459.7  18866.000  25294.5  35710.25  48103.3  66471.95
EngineSize       1.8      2.375      3.0      3.90      4.6      5.30
Cylinders        4.0      4.000      6.0      6.00      8.0      8.00
Horsepower     130.0    165.000    210.0    255.00    302.0    338.25
MPG_City        15.0     17.000     19.0     21.25     26.0     29.00
MPG_Highway     20.0     24.000     26.0     29.00     33.3     36.00
Weight        2678.7   3104.000   3474.5   3977.75   4494.4   4995.45
Wheelbase       99.0    103.000    107.0    112.00    119.0    123.00
Length         168.0    178.000    187.0    194.00    204.0    212.00

                  99%       max
```

```
MSRP          93659.00  192465.0
Invoice       87244.27  173560.0
EngineSize        6.00       8.3
Cylinders         9.50      12.0
Horsepower      469.71     500.0
MPG_City         35.73      60.0
MPG_Highway      43.73      66.0
Weight         5824.73    7190.0
Wheelbase       133.00     144.0
Length          222.00     238.0
```

[43]:
```
with pd.option_context('display.max_rows', 5):
    display(cars.describe(include=[np.number],
                percentiles=[.01, .05, .10, .25, .5, .75, .9, .95, .99]).T)
```

```
           count          mean           std      min       1%       5%  \
MSRP       428.0  32774.855140  19431.716674  10280.0  11191.45  13691.00
Invoice    428.0  30014.700935  17642.117750   9875.0  10659.01  12836.65
…            …           …            …        …        …
Wheelbase  428.0    108.154206      8.311813     89.0     93.00     95.35
Length     428.0    186.362150     14.357991    143.0    153.27    163.00

               10%       25%      50%       75%      90%       95%       99%  \
MSRP       15484.5  20334.25  27635.0  39205.00  52781.0  72864.25  93659.00
Invoice    14459.7  18866.00  25294.5  35710.25  48103.3  66471.95  87244.27
…              …        …       …        …       …        …        …
Wheelbase     99.0    103.00    107.0    112.00    119.0    123.00    133.00
Length       168.0    178.00    187.0    194.00    204.0    212.00    222.00

                max
MSRP       192465.0
Invoice    173560.0
…               …
Wheelbase     144.0
Length        238.0

[10 rows x 14 columns]
```

[45]:
```
cars.groupby(['Origin'])['EngineSize'].describe().round()
```

[45]:
```
        count  mean  std  min  25%  50%  75%  max
Origin
Asia    158.0   3.0  1.0  1.0  2.0  3.0  4.0  6.0
Europe  123.0   3.0  1.0  2.0  2.0  3.0  4.0  6.0
USA     147.0   4.0  1.0  2.0  3.0  4.0  5.0  8.0
```

[46]:
```
cars.groupby(['Origin'])[['EngineSize', 'MPG_City']].describe().round()
```

[46]:

```
       EngineSize                                          MPG_City              \
            count mean  std  min  25%  50%  75%  max     count  mean  std
Origin
Asia        158.0  3.0  1.0  1.0  2.0  3.0  4.0  6.0     158.0  22.0  7.0
Europe      123.0  3.0  1.0  2.0  2.0  3.0  4.0  6.0     123.0  19.0  3.0
USA         147.0  4.0  1.0  2.0  3.0  4.0  5.0  8.0     147.0  19.0  4.0


         min   25%   50%   75%   max
Origin
Asia    13.0  18.0  20.0  24.0  60.0
Europe  12.0  17.0  19.0  20.0  38.0
USA     10.0  17.0  18.0  21.0  29.0
```

[47]:

```python
cars.groupby(['Origin','Make'])[['EngineSize', 'MPG_City']].describe().round()
```

[47]:

```
                     EngineSize                                          MPG_City  \
                          count mean  std  min  25%  50%  75%  max      count
Origin Make
Asia   Acura                7.0  3.0  1.0  2.0  3.0  3.0  4.0  4.0        7.0
       Honda               17.0  2.0  1.0  1.0  2.0  2.0  3.0  4.0       17.0
       Hyundai             12.0  2.0  1.0  2.0  2.0  2.0  3.0  4.0       12.0
       Infiniti             8.0  4.0  1.0  4.0  4.0  4.0  4.0  4.0        8.0
       Isuzu                2.0  4.0  1.0  3.0  3.0  4.0  4.0  4.0        2.0
       Kia                 11.0  2.0  1.0  2.0  2.0  2.0  3.0  4.0       11.0
       Lexus               11.0  4.0  1.0  3.0  3.0  3.0  4.0  5.0       11.0
       Mazda               11.0  2.0  1.0  1.0  2.0  2.0  2.0  4.0       11.0
       Mitsubishi          13.0  3.0  1.0  2.0  2.0  2.0  4.0  4.0       13.0
       Nissan              17.0  3.0  1.0  2.0  3.0  4.0  4.0  6.0       17.0
       Scion                2.0  2.0  0.0  2.0  2.0  2.0  2.0  2.0        2.0
       Subaru              11.0  3.0  0.0  2.0  2.0  2.0  2.0  3.0       11.0
       Suzuki               8.0  2.0  0.0  2.0  2.0  2.0  2.0  3.0        8.0
       Toyota              28.0  3.0  1.0  2.0  2.0  3.0  3.0  5.0       28.0
Europe Audi                19.0  3.0  1.0  2.0  3.0  3.0  4.0  4.0       19.0
       BMW                 20.0  3.0  1.0  2.0  2.0  3.0  3.0  4.0       20.0
       Jaguar              12.0  4.0  1.0  2.0  4.0  4.0  4.0  4.0       12.0
       Land Rover           3.0  4.0  1.0  2.0  3.0  4.0  4.0  5.0        3.0
       MINI                 2.0  2.0  0.0  2.0  2.0  2.0  2.0  2.0        2.0
       Mercedes-Benz       26.0  4.0  1.0  2.0  3.0  3.0  5.0  6.0       26.0
       Porsche              7.0  4.0  1.0  3.0  3.0  4.0  4.0  4.0        7.0
       Saab                 7.0  2.0  0.0  2.0  2.0  2.0  2.0  2.0        7.0
       Volkswagen          15.0  3.0  1.0  2.0  2.0  2.0  4.0  6.0       15.0
       Volvo               12.0  2.0  0.0  2.0  2.0  2.0  3.0  3.0       12.0
USA    Buick                9.0  4.0  0.0  3.0  4.0  4.0  4.0  4.0        9.0
       Cadillac             8.0  5.0  1.0  4.0  5.0  5.0  5.0  6.0        8.0
       Chevrolet           27.0  4.0  1.0  2.0  3.0  4.0  5.0  6.0       27.0
       Chrysler            15.0  3.0  1.0  2.0  2.0  3.0  4.0  4.0       15.0
```

Dodge	13.0	3.0	2.0	2.0	2.0	4.0	4.0	8.0	13.0
Ford	23.0	4.0	1.0	2.0	2.0	4.0	5.0	7.0	23.0
GMC	8.0	5.0	1.0	3.0	4.0	5.0	5.0	6.0	8.0
Hummer	1.0	6.0	NaN	6.0	6.0	6.0	6.0	6.0	1.0
Jeep	3.0	3.0	1.0	2.0	3.0	4.0	4.0	4.0	3.0
Lincoln	9.0	4.0	1.0	3.0	4.0	5.0	5.0	5.0	9.0
Mercury	9.0	4.0	1.0	3.0	3.0	4.0	5.0	5.0	9.0
Oldsmobile	3.0	3.0	1.0	2.0	3.0	3.0	3.0	3.0	3.0
Pontiac	11.0	3.0	1.0	2.0	3.0	3.0	4.0	6.0	11.0
Saturn	8.0	2.0	0.0	2.0	2.0	2.0	2.0	3.0	8.0

		mean	std	min	25%	50%	75%	max
Origin	Make							
Asia	Acura	19.0	3.0	17.0	18.0	18.0	21.0	24.0
	Honda	28.0	11.0	17.0	21.0	26.0	32.0	60.0
	Hyundai	23.0	5.0	17.0	19.0	23.0	27.0	29.0
	Infiniti	17.0	1.0	15.0	17.0	18.0	18.0	19.0
	Isuzu	16.0	1.0	15.0	16.0	16.0	16.0	17.0
	Kia	22.0	4.0	16.0	18.0	24.0	24.0	26.0
	Lexus	17.0	2.0	13.0	18.0	18.0	18.0	20.0
	Mazda	21.0	4.0	15.0	18.0	23.0	24.0	26.0
	Mitsubishi	21.0	3.0	15.0	18.0	21.0	25.0	25.0
	Nissan	20.0	4.0	13.0	17.0	20.0	21.0	28.0
	Scion	32.0	1.0	31.0	31.0	32.0	32.0	32.0
	Subaru	20.0	1.0	18.0	20.0	21.0	21.0	22.0
	Suzuki	22.0	3.0	18.0	20.0	23.0	24.0	25.0
	Toyota	24.0	9.0	13.0	19.0	22.0	30.0	59.0
Europe	Audi	18.0	2.0	14.0	17.0	18.0	20.0	23.0
	BMW	19.0	2.0	16.0	18.0	19.0	20.0	21.0
	Jaguar	18.0	1.0	16.0	17.0	18.0	18.0	18.0
	Land Rover	14.0	3.0	12.0	12.0	12.0	15.0	18.0
	MINI	26.0	2.0	25.0	26.0	26.0	27.0	28.0
	Mercedes-Benz	17.0	3.0	13.0	16.0	18.0	19.0	22.0
	Porsche	17.0	2.0	14.0	17.0	18.0	18.0	20.0
	Saab	20.0	1.0	19.0	20.0	21.0	21.0	21.0
	Volkswagen	21.0	6.0	12.0	18.0	22.0	24.0	38.0
	Volvo	20.0	2.0	15.0	20.0	20.0	20.0	22.0
USA	Buick	19.0	2.0	15.0	18.0	20.0	20.0	20.0
	Cadillac	16.0	2.0	13.0	16.0	18.0	18.0	18.0
	Chevrolet	20.0	5.0	13.0	16.0	19.0	22.0	28.0
	Chrysler	20.0	2.0	17.0	18.0	21.0	22.0	22.0
	Dodge	19.0	5.0	12.0	16.0	18.0	21.0	29.0
	Ford	19.0	5.0	10.0	17.0	18.0	22.0	27.0
	GMC	15.0	2.0	13.0	14.0	16.0	16.0	18.0
	Hummer	10.0	NaN	10.0	10.0	10.0	10.0	10.0
	Jeep	17.0	2.0	16.0	16.0	16.0	18.0	20.0

```
Lincoln         17.0   2.0  13.0  17.0  17.0  17.0  20.0
Mercury         18.0   1.0  16.0  17.0  17.0  19.0  20.0
Oldsmobile      21.0   3.0  19.0  20.0  20.0  22.0  24.0
Pontiac         21.0   4.0  16.0  18.0  20.0  22.0  29.0
Saturn          24.0   3.0  20.0  23.0  26.0  26.0  26.0
```

[ ]:
```
# Plotting the world map based on  occurrences
import pandas as pd
import plotly.express as px
country_counts = cars['Origin'].value_counts().reset_index()
country_counts.columns = ['Country', 'count']
fig = px.choropleth(locations=country_counts['Country'], locationmode='country␣
↪names',
                    color=country_counts['count'],␣
↪hover_name=country_counts['Country'],
                    title='Occurrences by Country')
fig.update_layout(geo=dict(bgcolor='black'))
fig.show()
```

Heatmap

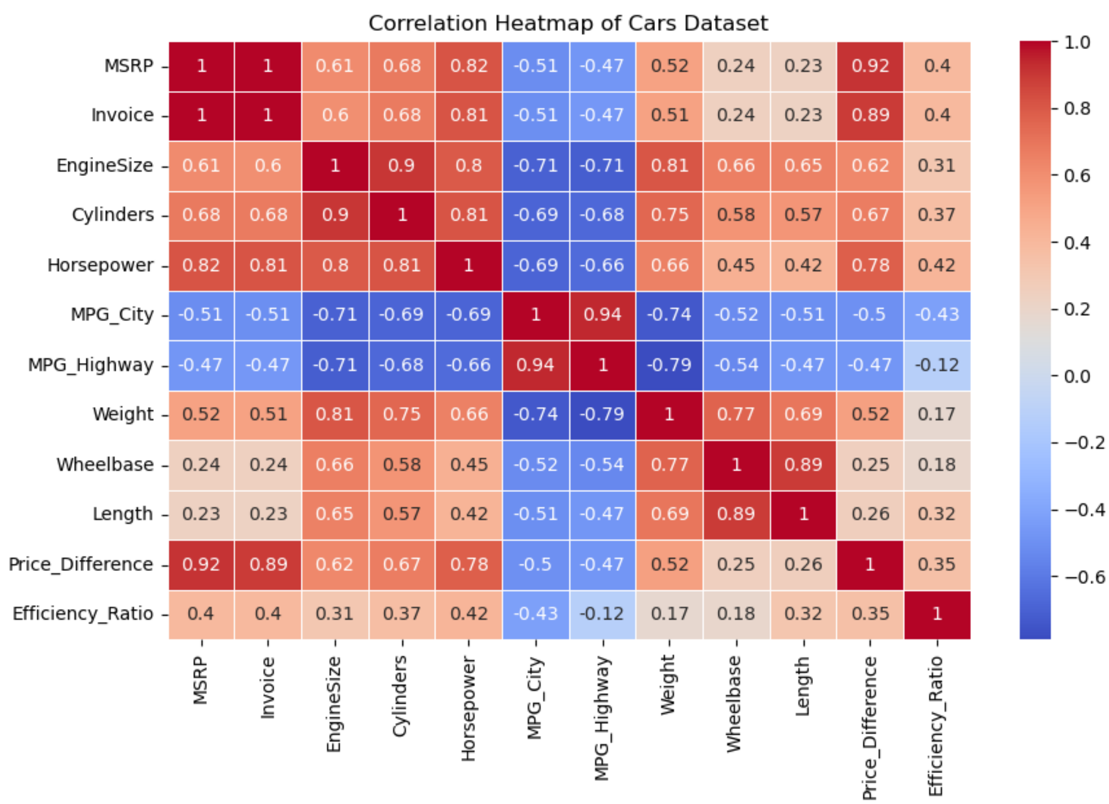
[49]:
```
corr = cars.select_dtypes(include=['number']).corr()
corr
```

[49]:
```
                       MSRP   Invoice  EngineSize  Cylinders  Horsepower  \
MSRP               1.000000  0.998856    0.609439   0.682940    0.817451
Invoice            0.998856  1.000000    0.601055   0.676663    0.813153
EngineSize         0.609439  0.601055    1.000000   0.904843    0.795363
Cylinders          0.682940  0.676663    0.904843   1.000000    0.811777
Horsepower         0.817451  0.813153    0.795363   0.811777    1.000000
MPG_City          -0.513148 -0.508727   -0.705414  -0.687004   -0.689460
MPG_Highway       -0.473169 -0.467997   -0.713757  -0.678195   -0.659110
Weight             0.516873  0.510650    0.808970   0.750218    0.656201
Wheelbase          0.242266  0.239036    0.657401   0.578381    0.446612
Length             0.232286  0.226808    0.651227   0.570997    0.422263
Price_Difference   0.915004  0.894665    0.622487   0.671266    0.776345
Efficiency_Ratio   0.395095  0.395436    0.314088   0.367509    0.416077

                   MPG_City  MPG_Highway    Weight  Wheelbase    Length  \
MSRP              -0.513148    -0.473169  0.516873   0.242266  0.232286
Invoice           -0.508727    -0.467997  0.510650   0.239036  0.226808
EngineSize        -0.705414    -0.713757  0.808970   0.657401  0.651227
Cylinders         -0.687004    -0.678195  0.750218   0.578381  0.570997
Horsepower        -0.689460    -0.659110  0.656201   0.446612  0.422263
MPG_City           1.000000     0.940326 -0.736972  -0.519170 -0.507673
MPG_Highway        0.940326     1.000000 -0.790348  -0.535909 -0.471433
Weight            -0.736972    -0.790348  1.000000   0.768463  0.693839
```

```
Wheelbase        -0.519170    -0.535909  0.768463   1.000000  0.889849
Length           -0.507673    -0.471433  0.693839   0.889849  1.000000
Price_Difference -0.501879    -0.472025  0.520459   0.246587  0.256519
Efficiency_Ratio -0.426060    -0.123128  0.168943   0.181847  0.316806

                  Price_Difference  Efficiency_Ratio
MSRP                      0.915004          0.395095
Invoice                   0.894665          0.395436
EngineSize                0.622487          0.314088
Cylinders                 0.671266          0.367509
Horsepower                0.776345          0.416077
MPG_City                 -0.501879         -0.426060
MPG_Highway              -0.472025         -0.123128
Weight                    0.520459          0.168943
Wheelbase                 0.246587          0.181847
Length                    0.256519          0.316806
Price_Difference          1.000000          0.354825
Efficiency_Ratio          0.354825          1.000000
```

[50]:

```python
plt.figure(figsize=(10,6))
sns.heatmap(corr, annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Correlation Heatmap of Cars Dataset")
plt.show()
```



18 Crosstab

[51]:
```
pd.crosstab(cars['Type'], cars['Origin'])
```

[51]:
```
Origin  asia  europe  usa
Type
hybrid     3       0    0
sedan     94      77   90
sports    17      20    9
suv       25      10   25
truck      8       0   16
wagon     11      12    7
```

[52]:
```
pd.crosstab(cars['DriveTrain'], cars['Origin'])
```

[52]:
```
Origin      asia  europe  usa
DriveTrain
all           34      36   22
front         99      37   90
rear          25      46   35
```

[53]:
```
pd.crosstab(cars['Type'], cars['Origin'], normalize='index') * 100
```

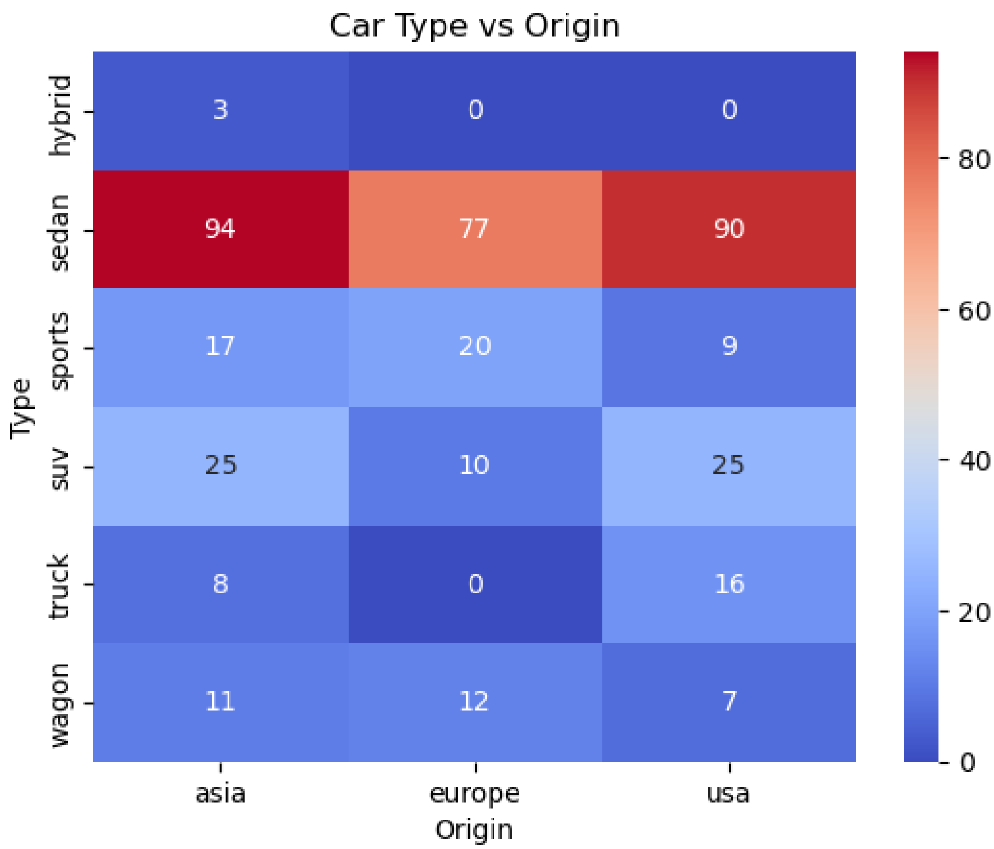
[53]:
```
Origin        asia     europe        usa
Type
hybrid  100.000000   0.000000   0.000000
sedan    36.015326  29.501916  34.482759
sports   36.956522  43.478261  19.565217
suv      41.666667  16.666667  41.666667
truck    33.333333   0.000000  66.666667
wagon    36.666667  40.000000  23.333333
```

[54]:
```
cars.pivot_table(values='Horsepower', index='Origin', columns='Type',␣
↪aggfunc='mean')
```

[54]:
```
Type    hybrid       sedan      sports     suv    truck       wagon
Origin
asia      92.0  181.978723  225.352941  214.16  190.250  185.636364
europe     NaN  233.194805  291.100000  263.10      NaN  218.166667
usa        NaN  191.988889  312.000000  246.56  242.125  165.714286
```

[56]:
```
ct = pd.crosstab(cars['Type'], cars['Origin'])
sns.heatmap(ct, annot=True, cmap="coolwarm", fmt="d")
plt.title("Car Type vs Origin")
```

```
plt.show()
```



18.1 Insights & Conclusions

Area	Key Findings
Fuel Efficiency & Weight	Heavier cars tend to have lower MPG in both city and highway driving.
Horsepower & Cylinders	Most cars fall between 4–6 cylinders with horsepower clustering around 200–250 HP.
Pricing Patterns	MSRP ranges from $10K to nearly $193K; median price is around $27K. Luxury cars form high-end outliers.
Regional Differences	Asian cars are lighter and more fuel-efficient; European cars have higher horsepower and premium pricing; US cars typically have larger engines and heavier builds.

Area	Key Findings
DriveTrain Trends	Rear-wheel drive is common in performance/luxury cars, while front-wheel drive dominates economy cars.
Correlation Insights	Strong negative correlation between Weight and MPG; positive correlation between Horsepower and Weight.

Conclusion:
The Cars dataset highlights clear patterns in performance, efficiency, and pricing. Vehicle specifications such as weight, horsepower, and engine size strongly influence fuel efficiency and cost. Regional differences also show distinct design philosophies across USA, Asia, and Europe.

[ ]:

```

```