

```
1 import numpy as np
2 import pandas as pd
3 import matplotlib.pyplot as plt
4 import matplotlib as mplt
5 import random
6 from math import *
7 import sklearn as svm
8 import seaborn as sns
9 import plotly as pl
10 import plotly.express as px
11 import plotly.graph_objects as go
12 from plotly.subplots import make_subplots
13 import matplotlib.ticker as plticker
14 %matplotlib inline
```

Car prices dataset analysis

1. Pre data analysis.

Reading dataset and convertign into a dataframe

```
1 data = pd.read_csv("/content/car_prices_dataset.csv")

1 dataframe = pd.DataFrame(data)
2 dataframe.head()
```

	year	make	model	trim	body	transmission	vin	state	condition	odometer	color	interior	seller	
0	2015	Kia	Sorento	LX	SUV	automatic	5xyktca69fg566472	ca	5.0	16639.0	white	black	kia motors america inc	205C
1	2015	Kia	Sorento	LX	SUV	automatic	5xyktca69fg561319	ca	5.0	9393.0	white	beige	kia motors america inc	208C
2	2014	BMW	3 Series	328i SULEV	Sedan	automatic	wba3c1c51ek116351	ca	45.0	1331.0	gray	black	financial services remarketing (lease)	319C
3	2015	Volvo	S60	T5	Sedan	automatic	yv1612tb4f1310987	ca	41.0	14282.0	white	black	volvo na rep/world omni	275C
4	2014	BMW	6 Series Gran Coupe	650i	Sedan	automatic	wba6b2c57ed129731	ca	43.0	2641.0	gray	black	financial services remarketing (lease)	660C

Let us now fill all the missing of NaN values, so that we do not have any issues later on.

```
1 dataframe.dropna(inplace=True)
```

Let us now perform some inspective data analysis. In order to do so, we will persom some pandas funcions such as: info, describe.

```
1 # Pandas info function
2 dataframe.info()
3
```

```
4 # Pandas describe function
5 dataframe.describe()
```

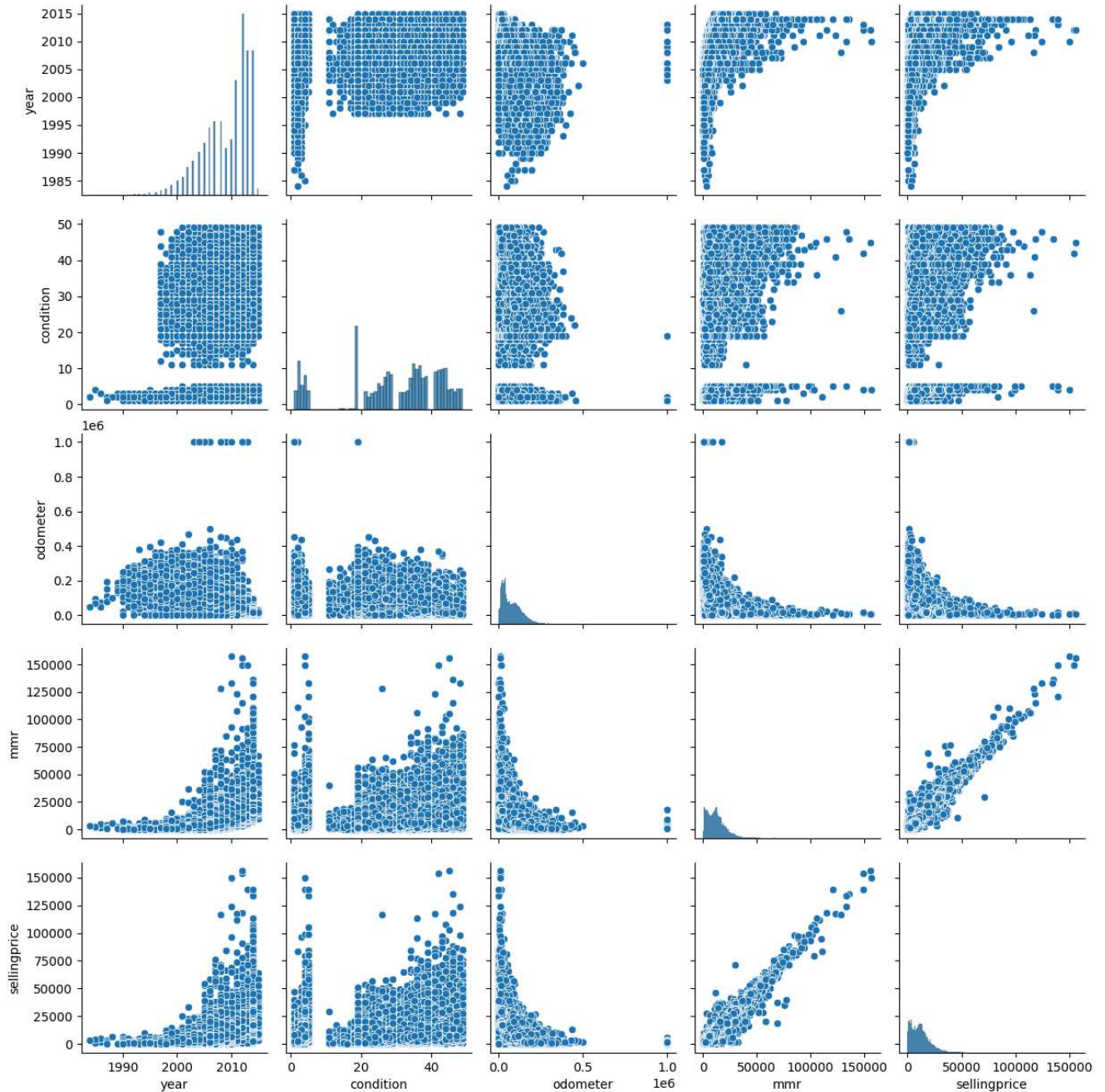
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 558837 entries, 0 to 558836
Data columns (total 16 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   year            558837 non-null  int64
1   make            558837 non-null  object
2   model           558837 non-null  object
3   trim            558837 non-null  object
4   body            558837 non-null  object
5   transmission    558837 non-null  object
6   vin             558837 non-null  object
7   state           558837 non-null  object
8   condition       558837 non-null  float64
9   odometer        558837 non-null  float64
10  color           558837 non-null  object
11  interior         558837 non-null  object
12  seller          558837 non-null  object
13  mmr             558837 non-null  float64
14  sellingprice    558837 non-null  float64
15  saledate        558837 non-null  object
dtypes: float64(4), int64(1), object(11)
memory usage: 68.2+ MB
```

	year	condition	odometer	mmr	sellingprice
count	558837.000000	558837.000000	558837.000000	558837.000000	558837.000000
mean	2010.038927	30.023612	68308.525898	13768.441200	13611.066531
std	3.966864	13.975491	53401.402078	9680.303934	9749.600973
min	1982.000000	0.000000	0.000000	0.000000	0.000000
25%	2007.000000	22.000000	28359.000000	7100.000000	6900.000000
50%	2012.000000	34.000000	52245.000000	12250.000000	12100.000000
75%	2013.000000	41.000000	99103.000000	18300.000000	18200.000000
max	2015.000000	49.000000	999999.000000	182000.000000	230000.000000

Having basic data, we can have a general visualization of how the columns/values look together. It would be easier for us later on, to see what further analysis we can make.


```
1 sns.pairplot(dataframe)
```

 <seaborn.axisgrid.PairGrid at 0x7aab07c21ae0>



Lets check all the columns, so we can know what data can we compare, and what will be the best comparisons.

```
1 columns = dataframe.columns
2 columns
```

 Index(['year', 'make', 'model', 'trim', 'body', 'transmission', 'vin', 'state',  
'condition', 'odometer', 'color', 'interior', 'seller', 'mmr',  
'sellingprice', 'saledate'],  
dtype='object')

## 2. In depth data analysis of self chosen values.

## 2.1 Brand (make) vs Total selling price for that brand. Lets see what brands are making the most.

### A. Scatter plots.

#### Plotly scattter plot.

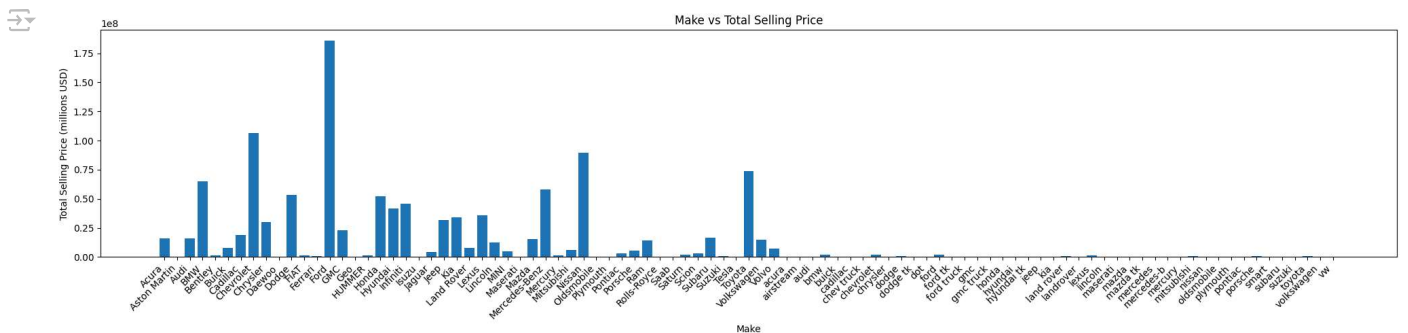
```
1 px.scatter(dataframe, x='make', y='sellingprice', title='Make vs Total Selling Price')
```



### B. Histograms

#### Matplotlib histogram

```
1 make_total_selling_price = dataframe.groupby('make')['sellingprice'].sum()
2
3 plt.figure(figsize=(20, 5))
4 plt.bar(make_total_selling_price.index, make_total_selling_price.values)
5 plt.xlabel('Make')
6 plt.ylabel('Total Selling Price (millions USD)')
7 plt.title('Make vs Total Selling Price')
8 plt.xticks(rotation=45, ha='right')
9 plt.tight_layout()
10 plt.show()
```



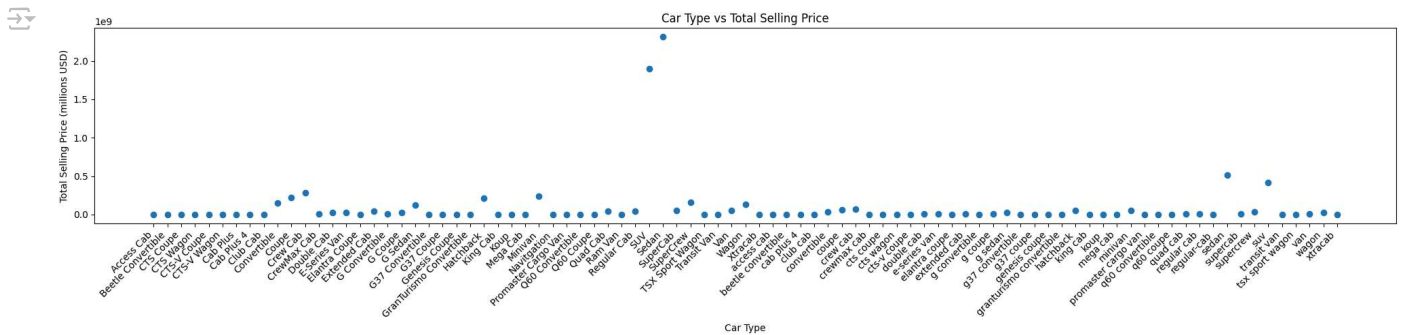
#### Plotly histogram

```
1 px.histogram(dataframe, x='make', y='sellingprice', title='Make vs Total Selling Price')
```

## 2.2 Comparison between body (car type) and seelingprice

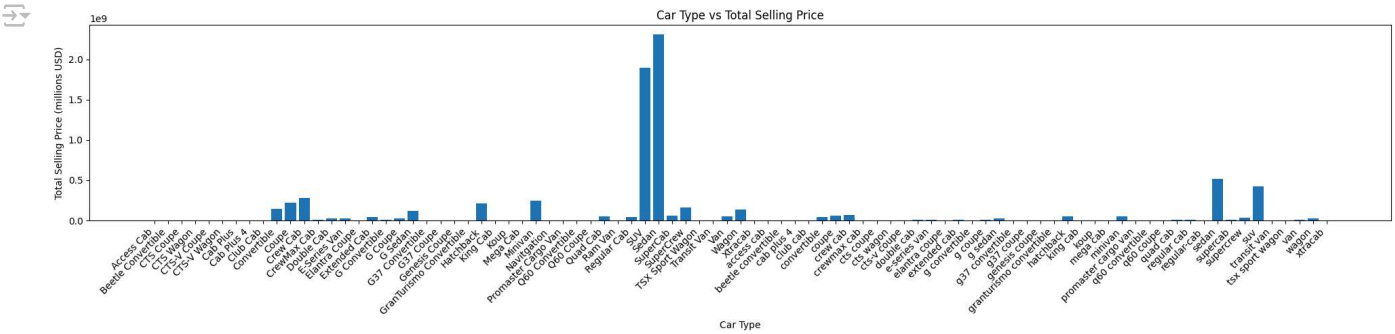
## Matplotlib scatter plot

```
1 total_selling_price_by_car_type = dataframe.groupby('body')['sellingprice'].sum()
2 figure = plt.figure(figsize=(20, 5))
3 plt.scatter(total_selling_price_by_car_type.index, total_selling_price_by_car_type.values)
4 plt.xlabel('Car Type')
5 plt.ylabel('Total Selling Price (millions USD)')
6 plt.title('Car Type vs Total Selling Price')
7 plt.xticks(rotation=45, ha='right')
8 plt.tight_layout()
9 plt.show()
```



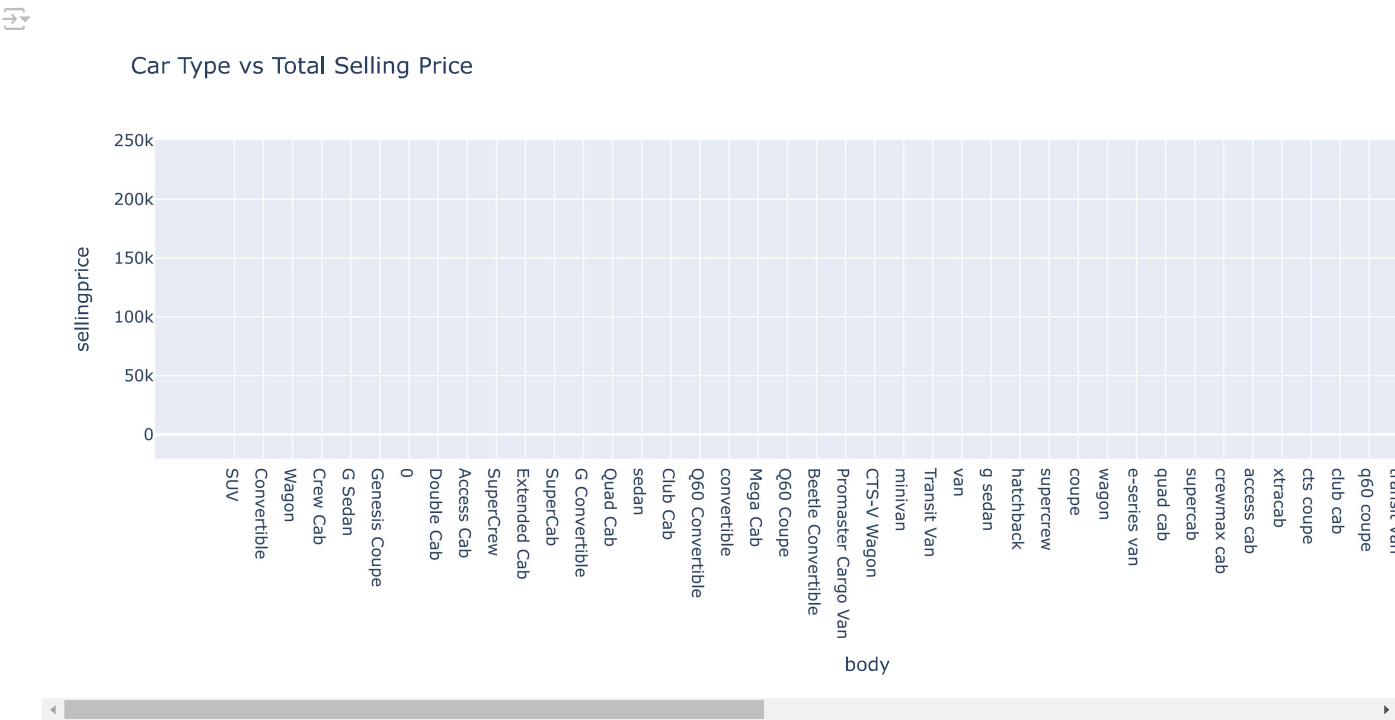
## 2. Matplotlib histogram plot

```
1 total_selling_price_by_car_type = dataframe.groupby('body')['sellingprice'].sum()
2 figure = plt.figure(figsize=(20, 5))
3 plt.bar(total_selling_price_by_car_type.index, total_selling_price_by_car_type.values)
4 plt.xlabel('Car Type')
5 plt.ylabel('Total Selling Price (millions USD)')
6 plt.title('Car Type vs Total Selling Price')
7 plt.xticks(rotation=45, ha='right')
8 plt.tight_layout()
9 plt.show()
```



Plotly scatter plot.

```
1 px.scatter(dataframe, x='body', y='sellingprice', title='Car Type vs Total Selling Price')
```

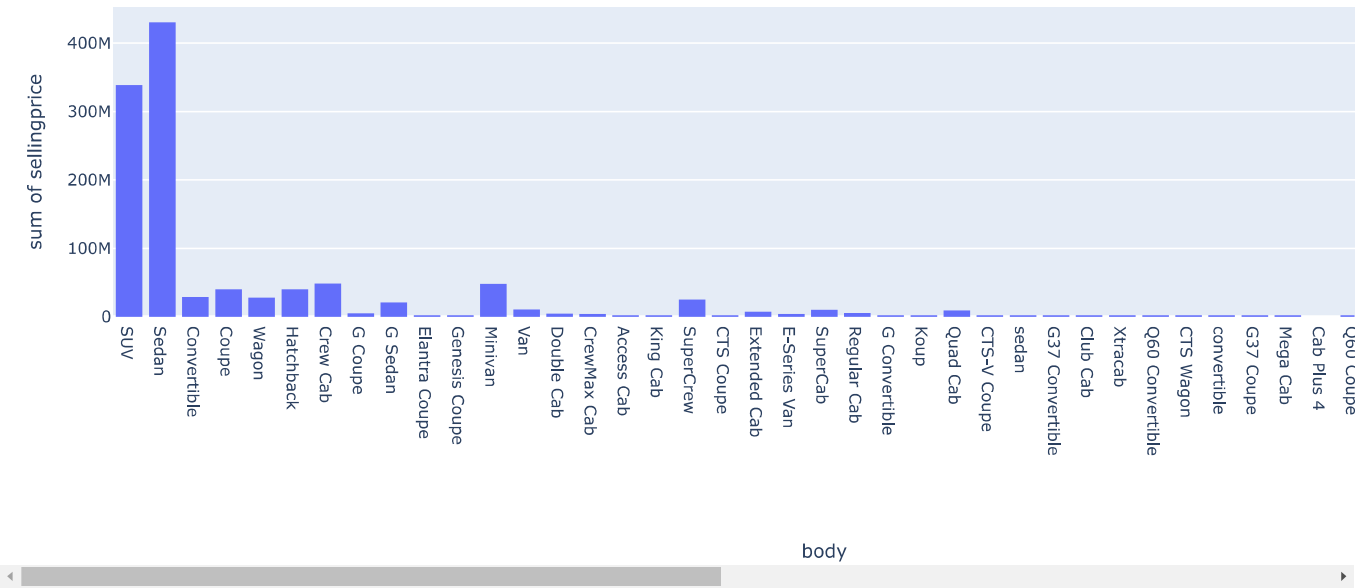


Plotly histogram

```
1 px.histogram(dataframe, x='body', y='sellingprice', title='Car Type vs Total Selling Price')
```



Car Type vs Total Selling Price



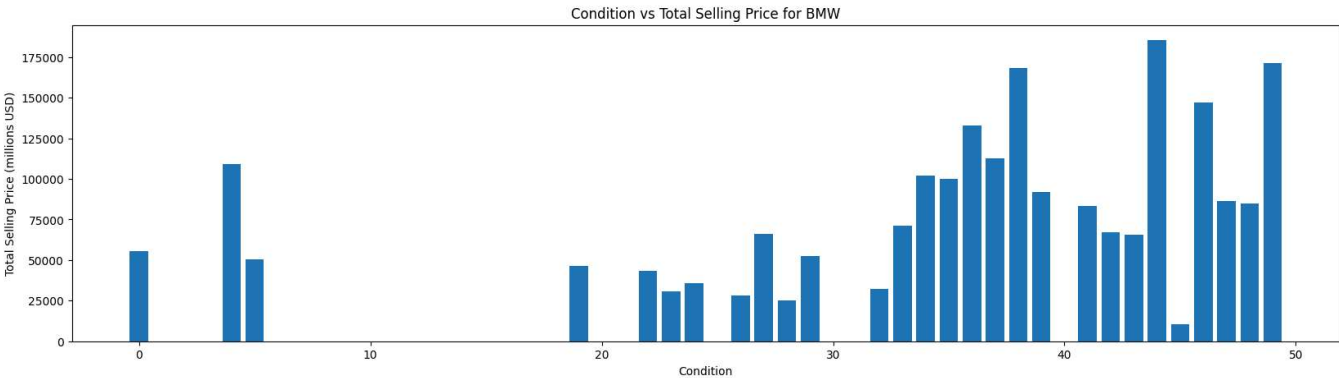
2.3 Condition vs total selling price for BMW.

A. Histograms

Matplotlib

```
1 condition_total_selling_price_bmw = dataframe[dataframe['make'] == 'bmw'].groupby('condition')['sellingprice'].sum()
2
3 plt.figure(figsize=(20, 5))
4 plt.bar(condition_total_selling_price_bmw.index, condition_total_selling_price_bmw.values)
5 plt.xlabel('Condition')
6 plt.ylabel('Total Selling Price (millions USD)')
7 plt.title('Condition vs Total Selling Price for BMW')

plt.show()
```

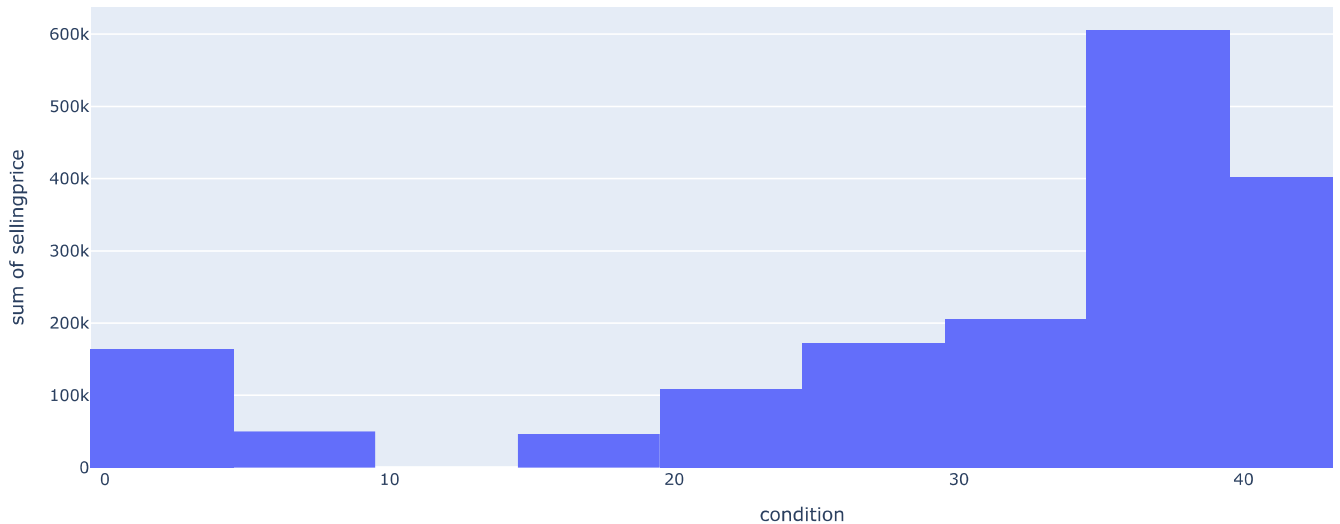


Plotly

```
1 px.histogram(dataframe[dataframe['make'] == 'bmw'], x='condition', y='sellingprice', title='Condition vs Total Selling Price for BMW')
```



### Condition vs Total Selling Price for BMW



### B. Line plots

#### Matplotlib

```

1 condition_total_selling_price_bmw = dataframe[dataframe['make'] == 'bmw'].groupby('condition')['sellingprice'].sum()
2
3 plt.figure(figsize=(20, 5))
4 plt.plot(condition_total_selling_price_bmw.index, condition_total_selling_price_bmw.values, marker = "*",
5 markersize = 10)
6 plt.xlabel('Condition')
7 plt.ylabel('Total Selling Price (millions USD)')
8 plt.title('Condition vs Total Selling Price for BMW')

```



Text(0.5, 1.0, 'Condition vs Total Selling Price for BMW')



#### Plotly

```

1 # Filter data for BMW cars
2 bmw_data = dataframe[dataframe['make'] == 'bmw']
3
4 # Group by 'condition' and calculate total selling price
5 condition_selling_price = bmw_data.groupby('condition')['sellingprice'].sum().reset_index()
6
7 # Create the line plot with markers
8 fig = px.line(
9     condition_selling_price,
10    x='condition',
11    y='sellingprice',
12    title='Condition vs Total Selling Price for BMW',
13    markers=True # Show markers on the line

```



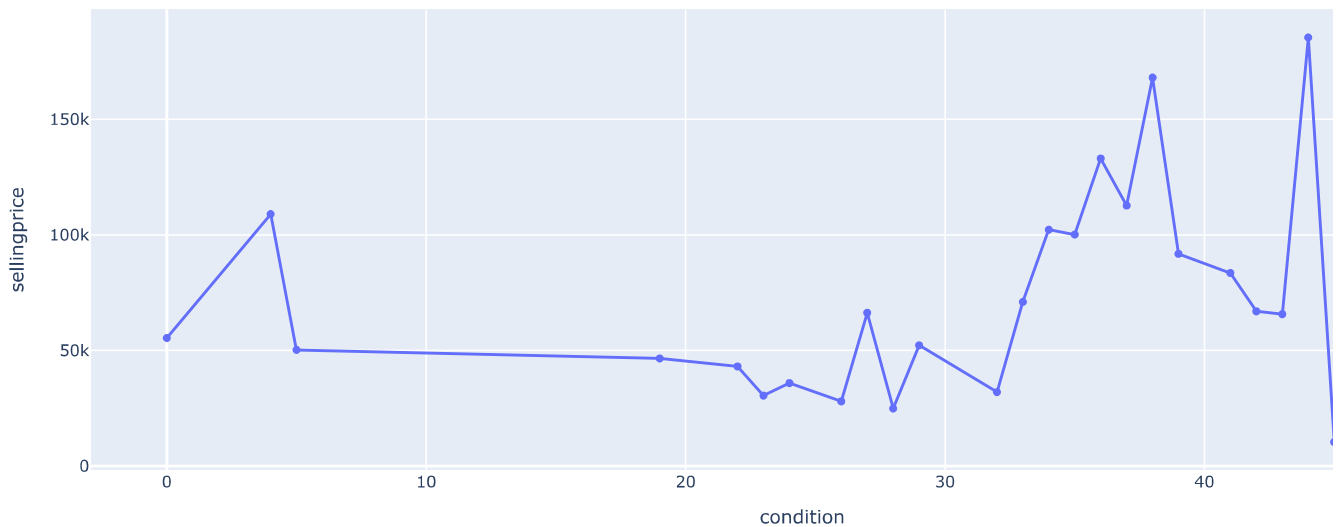
```

14 )
15
16 # Show the plot
17 fig.show()

```



Condition vs Total Selling Price for BMW



### 3. Pie charts.

For the final part of the charts lets use pie charts. For this we will have to choose the data for which we want to see what part of the whole it makes. As we have lot's of manufacturers and car types and it would look messy on a pie chart, we are going to use a part of the data for it.

First, lets get a reminder of the data/columns that we have in our dataframe

```

1 columns = dataframe.columns
2 columns

Index(['year', 'make', 'model', 'trim', 'body', 'transmission', 'vin', 'state',
      'condition', 'odometer', 'color', 'interior', 'seller', 'mmr',
      'sellingprice', 'saledate'],
      dtype='object')

```

**3.1 Marketshare by car brands.Make (car brand) vs sellingprice (for better visualization we are dropping all values where the marketshare is less than 3%)**

#### Matplotlib

```

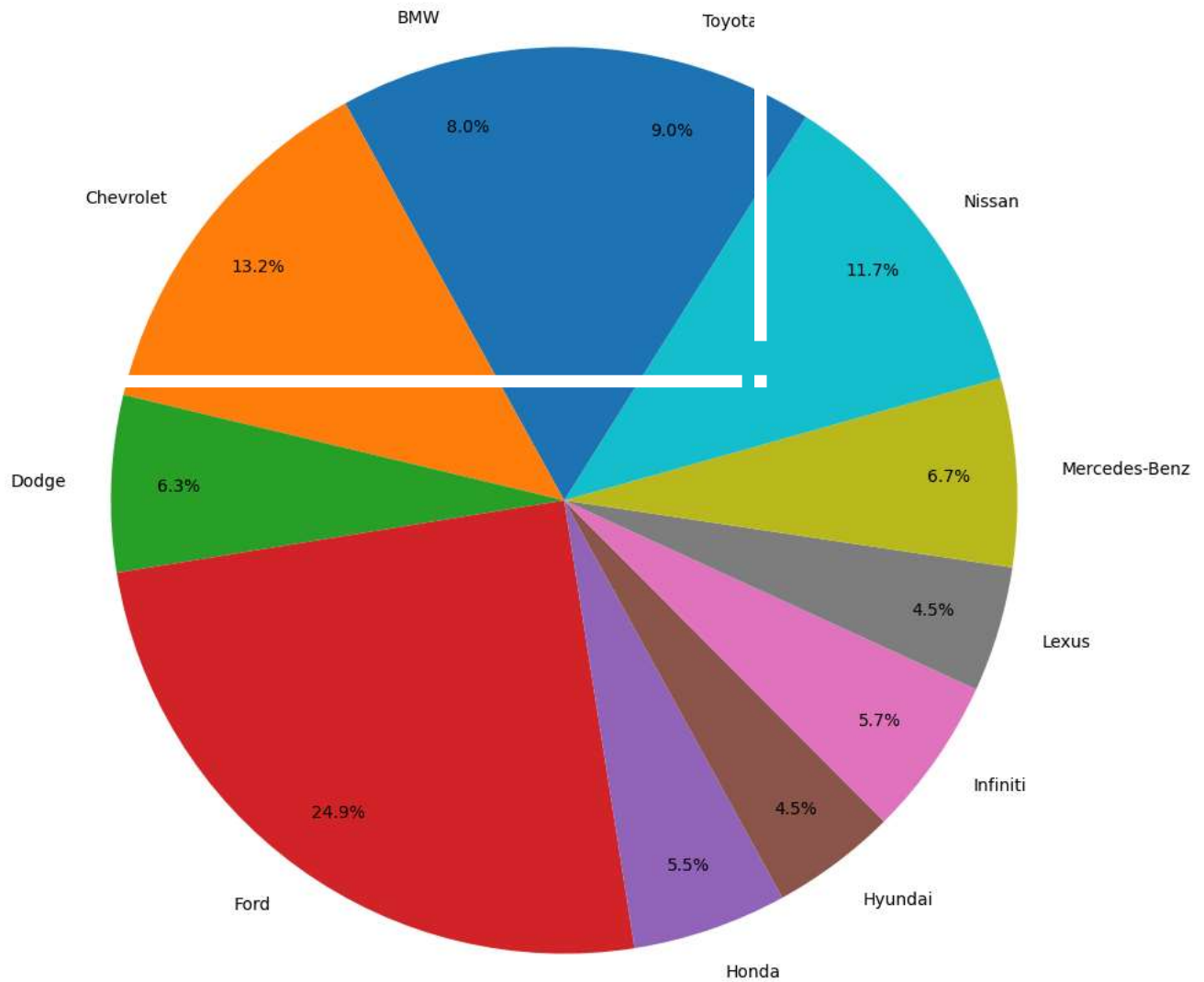
1 # Sample the dataframe (1/5 of the data)
2 sampled_df = dataframe.sample(frac=0.2, random_state=42)
3
4 # Calculate total selling price for each car make
5 car_brands_market_share = sampled_df.groupby('make')['sellingprice'].sum()
6
7 # Calculate market share percentage for each car make
8 total_selling_price = car_brands_market_share.sum()
9 car_brands_market_share = (car_brands_market_share / total_selling_price) * 100
10
11 # Filter out market shares below 3%
12 car_brands_market_share = car_brands_market_share[car_brands_market_share >= 3]
13
14 # Create the pie chart
15 plt.figure(figsize=(12, 12)) # Adjust figure size as needed
16 plt.pie(car_brands_market_share, labels=car_brands_market_share.index,
17         autopct='%1.1f%%', startangle=90,
18         pctdistance=0.85, labeldistance=1.1)
19

```

```
20 plt.title('Marketshare by Car Brands (Sampled Data, Market Share >= 3%)')
```



Marketshare by Car Brands (Sampled Data, Market Share &gt;= 3%)



### Plotly

```
1 sampled_df = dataframe.sample(frac=0.2, random_state=42)
2
3 # Group by 'make' and calculate total selling price for each make
4 car_brands_market_share = sampled_df.groupby('make')['sellingprice'].sum().reset_index()
5
6 # Calculate market share percentage
7 total_selling_price = car_brands_market_share['sellingprice'].sum()
8 car_brands_market_share['market_share_percentage'] = (car_brands_market_share['sellingprice'] / total_selling_price) * 100
9
10 # Filter out market shares below 3%
```