te8rpjycz

January 25, 2025

[4]:
```
"""Title: Bike Details Dataset"""
```

[4]:
```
'Title: Bike Details Dataset'
```

[5]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[7]:
```
df = pd.read_csv('BIKE DETAILS.csv')
```

[8]:
```
df.head()
```

[8]:

	name	selling_price	year	seller_type	owner	km_driven	ex_showroom_price
0	Royal Enfield Classic 350	175000	2019	Individual	1st owner	350	NaN
1	Honda Dio	45000	2017	Individual	1st owner	5650	NaN
2	Royal Enfield Classic Gunmetal Grey	150000	2018	Individual	1st owner	12000	148114.0
3	Yamaha Fazer FI V 2.0 [2016-2018]	65000	2015	Individual	1st owner	23000	89643.0
4	Yamaha SZ [2013-2014]	20000	2011	Individual	2nd owner	21000	NaN

[9]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1061 entries, 0 to 1060
Data columns (total 7 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   name               1061 non-null   object
 1   selling_price      1061 non-null   int64
 2   year               1061 non-null   int64
 3   seller_type        1061 non-null   object
```

```
 4   owner              1061 non-null   object
 5   km_driven          1061 non-null   int64
 6   ex_showroom_price  626 non-null    float64
dtypes: float64(1), int64(3), object(3)
memory usage: 58.1+ KB
```

[11]:
```
df.isnull().sum()
```

[11]:
```
name                  0
selling_price         0
year                  0
seller_type           0
owner                 0
km_driven             0
ex_showroom_price    435
dtype: int64
```

[13]:
```
df.dropna(inplace = True)
```

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

[14]:
```
name                 0
selling_price        0
year                 0
seller_type          0
owner                0
km_driven            0
ex_showroom_price    0
dtype: int64
```

##Q1. What is the range of selling prices in the dataset?

[17]:
```
#will do the same questions with 2 differ methods >>
selling_price_range = df['selling_price'].max() - df['selling_price'].min()

print(selling_price_range)
```

```
754000
```

[20]:
```
#2nd way >>
# Using NumPy's ptp method
column_range = np.ptp(df['selling_price'])

print("Range of selling_price:", column_range)
```

```
Range of selling_price: 754000
```

###Q2 What is the median selling price for bikes in the dataset?

[21]:
```
df['selling_price'].median()
```

[21]:
```
np.float64(45000.0)
```

Q3. What is the most common seller type?

[24]:
```
df['seller_type'].mode()[0] #used zero to extraxt first mode value if ties
```

[24]:
```
'Individual'
```

Q4. How many bikes have driven more than 50,000 kilometers?

[28]:
```
df[df['km_driven'] > 50000].count()
```

[28]:
```
name                88
selling_price       88
year                88
seller_type         88
owner               88
km_driven           88
ex_showroom_price   88
dtype: int64
```

[29]:
```
#88 bikes are there which driven more than50000 kms
```

Q5. What is the average km_driven value for each ownership type?

[30]:
```
df.groupby('owner')['km_driven'].mean()
```

[30]:
```
owner
1st owner    33067.926259
2nd owner    29809.484848
3rd owner    30904.666667
4th owner     6500.000000
Name: km_driven, dtype: float64
```

Q6. What proportion of bikes are from the year 2015 or older?

[32]:
```
df.head() #to check the dataset
```

[32]:
```
                                name  selling_price  year seller_type  \
2  Royal Enfield Classic Gunmetal Grey         150000  2018  Individual
3    Yamaha Fazer FI V 2.0 [2016-2018]          65000  2015  Individual
5                    Honda CB Twister          18000  2010  Individual
6                Honda CB Hornet 160R          78500  2018  Individual
9                  Bajaj Discover 125          50000  2016  Individual

       owner  km_driven  ex_showroom_price
```

```
2   1st owner      12000          148114.0
3   1st owner      23000           89643.0
5   1st owner      60000           53857.0
6   1st owner      17000           87719.0
9   1st owner      42000           60122.0
```

[45]:
```
bikes_2015_or_older  = df[df['year'] <= 2015].shape[0]
```

[46]:
```
#to find proportion will devide >> df[df['year'] >= 2015].shape[0] / Total␣
 ↪number of bikes

prop = bikes_2015_or_older / df.shape[0]
print(f'The proportion of bikes from 2015 or older is:', prop)
```

```
The proportion of bikes from 2015 or older is: 0.5287539936102237
```

Q. What is the trend of missing values across the dataset?

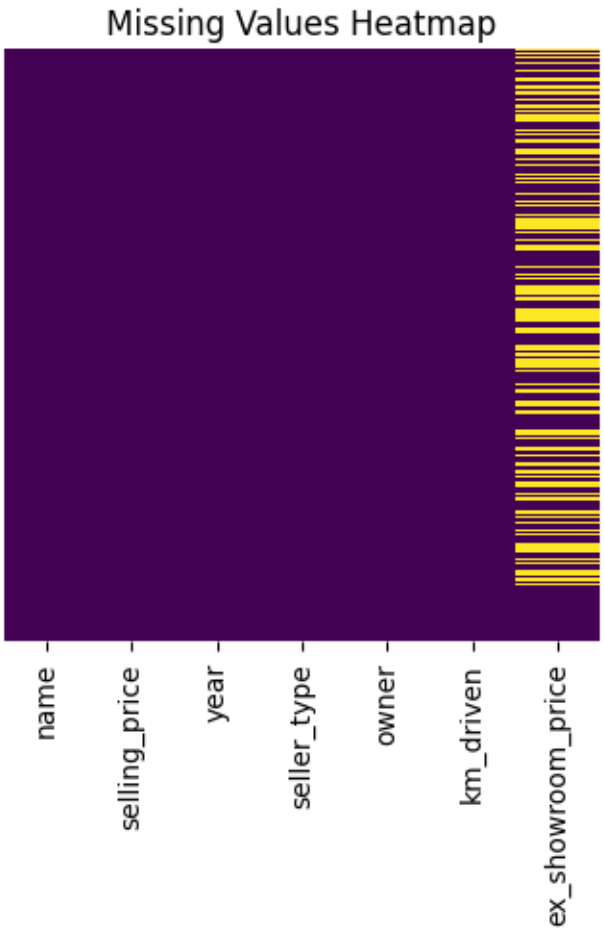
[47]:
```
#to check the trend we need to again import the data but with another name >>

null = pd.read_csv('BIKE DETAILS.csv')

null.isnull().sum()
```

[47]:
```
name                  0
selling_price         0
year                  0
seller_type           0
owner                 0
km_driven             0
ex_showroom_price    435
dtype: int64
```

[51]:
```
#above shown that there are no null values in columns expect ex_showroom_price.
#to visualising it >>
plt.figure(figsize=(4, 4))
sns.heatmap(null.isnull(), cbar=False, cmap="viridis", yticklabels=False)
plt.title("Missing Values Heatmap")
plt.show()
```



[52]:
```
#insight >> from above heatmap visualisation we have got to know that maximum␣
↪null values are found in only one column i.e., ex_showroom_price
```

Q8. What is the highest ex_showroom_price recorded, and for which bike?

[60]:
```
max_p = df[df['ex_showroom_price'] == df['ex_showroom_price'].max()]

max_p #pull out the row with maximum price
```

[60]:
```
                          name  selling_price  year seller_type      owner  \
134  Harley-Davidson Street Bob         750000  2013  Individual  2nd owner

     km_driven  ex_showroom_price
134      12000          1278000.0
```

[69]:
```
Highest_price = max_p['ex_showroom_price']
Name = max_p['name'].iloc[0]
```

```
print(f"The bike {Name}, and has the highest ex_showroom_price i.e.,␣
  ↪{Highest_price}")
```

```
The bike Harley-Davidson Street Bob, and has the highest ex_showroom_price i.e.,
134    1278000.0
Name: ex_showroom_price, dtype: float64
```

Q9. What is the total number of bikes listed by each seller type?

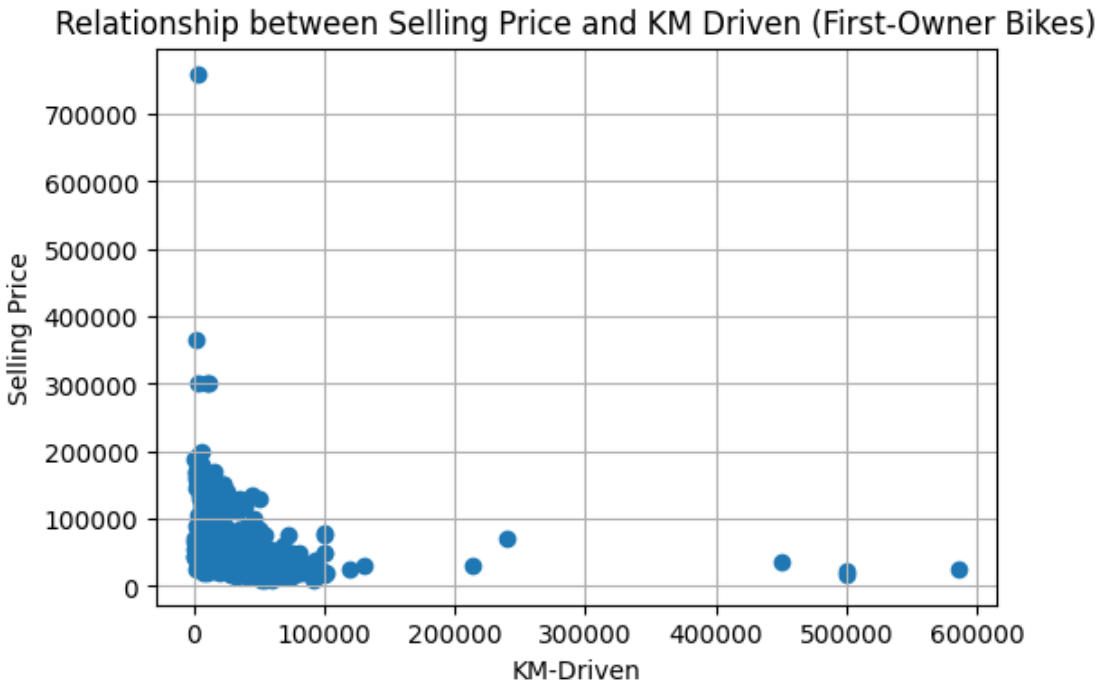
[72]:
```
df.groupby('seller_type')['name'].count()
```

[72]:
```
seller_type
Dealer          3
Individual    623
Name: name, dtype: int64
```

Q10.What is the relationship between selling_price and km_driven for first-owner bikes?

[75]:
```
first_own = df[df['owner'] == '1st owner']
```

[79]:
```
plt.figure(figsize = (6,4))
plt.scatter(first_own['km_driven'], first_own['selling_price'])
plt.title('Relationship between Selling Price and KM Driven (First-Owner␣
  ↪Bikes)')
plt.xlabel("KM-Driven")
plt.ylabel("Selling Price")
plt.grid(True)
plt.show()
```



[80]:
```
#insight >> from above table we got to know that >>
#1. Most data points are concentrated in the lower range of km_driven (below␣
 ↪100,000 km) and selling_price (below 100,000).
#2. This suggests that bikes with lower mileage generally have higher resale␣
 ↪value.
#3. A few points are far from the main cluster, such as bikes with very high␣
 ↪prices (above 400,000) or mileage above 300,000 km. These could represent␣
 ↪rare or premium models.
#4. The scatter suggests a negative correlation, where bikes with higher␣
 ↪mileage tend to have lower selling prices.
```

Q11. Identify and remove outliers in the km_driven column using the IQR method?

[83]:
```
#calculating IQR

Q1 = df['km_driven'].quantile(25/100)

Q3 = df['km_driven'].quantile(75/100)

IQR = Q3-Q1
print(IQR)
```

```
26968.75
```

[86]:
```
Lower_bound = Q1 - 1.5*IQR
Upper_bound = Q3 + 1.5*IQR

print(Lower_bound)
print(Upper_bound)
```

```
-27421.875
80453.125
```

[88]:
```
outliers = df[(df['km_driven'] < Lower_bound) | (df['km_driven'] > Upper_bound)]

outliers.shape[0]
```

[88]:
```
23
```

[89]:
```
df_cleaned = df[(df['km_driven'] >= Lower_bound) & (df['km_driven'] <=␣
 ↪Upper_bound)]
```

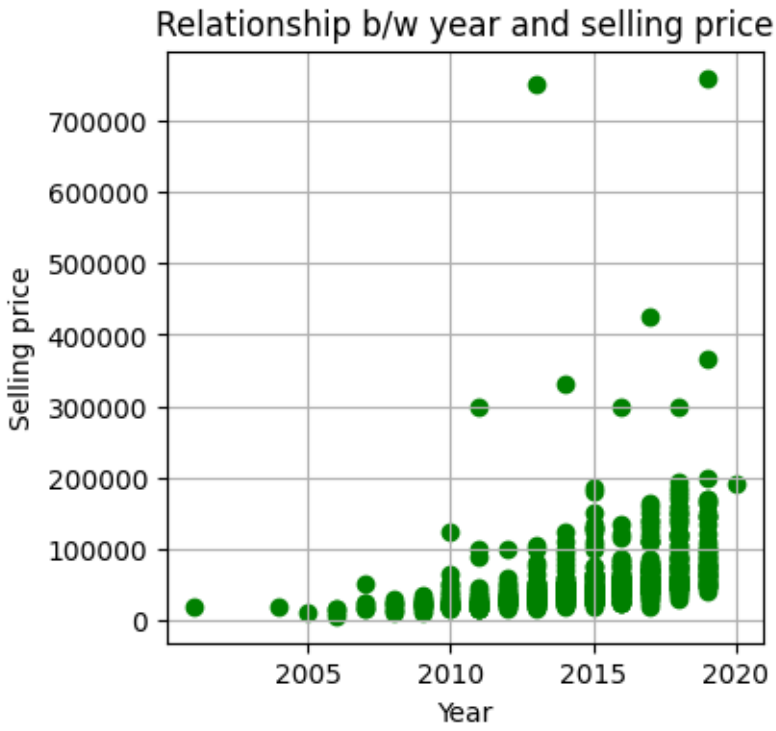
[92]:
```
df_cleaned.count()
```

[92]:
```
name                603
selling_price       603
year                603
seller_type         603
owner               603
km_driven           603
ex_showroom_price   603
dtype: int64
```

[93]:
```
#outliners removed in above data set df_cleaned.
```

Q12. Perform a bivariate analysis to visualize the relationship between year and selling_price?

[98]:
```
plt.figure(figsize = (4,4))
plt.scatter(df['year'], df["selling_price"], color = "green")
plt.title("Relationship b/w year and selling price")
plt.xlabel("Year")
plt.ylabel("Selling price")
plt.grid(True)
plt.show()
```



#insight >> from above plot we got to know that the cars that are older have less prices and the number of older cars are lesser than recent cars. Cars that have higher values are mostly come between 2015 to 2020

Q13. What is the average depreciation in selling price based on the bike's age (current year - manufacturing year)?

[102]:
```
df.head()
```

[102]:

	name	selling_price	year	seller_type	owner	km_driven	ex_showroom_price
2	Royal Enfield Classic Gunmetal Grey	150000	2018	Individual	1st owner	12000	148114.0
3	Yamaha Fazer FI V 2.0 [2016-2018]	65000	2015	Individual	1st owner	23000	89643.0
5	Honda CB Twister	18000	2010	Individual	1st owner	60000	53857.0
6	Honda CB Hornet 160R	78500	2018	Individual	1st owner	17000	87719.0
9	Bajaj Discover 125	50000	2016	Individual	1st owner	42000	60122.0

[104]:
```
from datetime import datetime

current_year = datetime.now().year

print(current_year)
```

2025

[105]:
```
#adding new column for bike's age

df['Age'] = current_year - df.year
```

[106]:
```
df.head()
```

[106]:

	name	selling_price	year	seller_type	owner	km_driven	ex_showroom_price	Age
2	Royal Enfield Classic Gunmetal Grey	150000	2018	Individual	1st owner	12000	148114.0	7
3	Yamaha Fazer FI V 2.0 [2016-2018]	65000	2015	Individual	1st owner	23000	89643.0	10
5	Honda CB Twister	18000	2010	Individual	1st owner	60000	53857.0	15
6	Honda CB Hornet 160R	78500	2018	Individual	1st owner	17000	87719.0	7
9	Bajaj Discover 125	50000	2016	Individual	1st owner	42000	60122.0	9

[108]:
```
#now we need to find the average of selling price on the basis age

avg_sp = df.groupby('Age')['selling_price'].mean()
print(avg_sp)
```

```
Age
5     190000.000000
6     111125.000000
7      87837.662338
8      70529.411765
9      53597.440476
10     54550.000000
11     49453.030303
12     52349.056604
13     32810.486486
14     36787.878788
15     32057.142857
16     22642.857143
17     19871.428571
18     24983.333333
19     11500.000000
20     10000.000000
21     18000.000000
24     20000.000000
Name: selling_price, dtype: float64
```

[112]:
```
initial_price = avg_sp.iloc[0] #location based indexing for selection by␣
↪position.
```

[115]:
```
avg_dep = (initial_price - avg_sp) / avg_sp.index
print(avg_dep)
```

```
Age
5         0.000000
6     13145.833333
7     14594.619666
8     14933.823529
9     15155.839947
10    13545.000000
11    12776.997245
12    11470.911950
13    12091.501040
14    10943.722944
15    10529.523810
16    10459.821429
17    10007.563025
18     9167.592593
19     9394.736842
```

```
20      9000.000000
21      8190.476190
24      7083.333333
dtype: float64
```

[116]:
```
print(f"Overall Average depericiation: {avg_dep.mean()}")
```

```
Overall Average depericiation: 10693.960937543588
```

#Insight >> From above observation we got to know that the overall average depreciation is 10693.96
#The average depreciation in selling price reflects how much the price of a bike decreases each year.

Q. Which bike names are priced significantly above the average price for their manufacturing year?

[118]:
```
avg_price_by_year = df.groupby("year")['selling_price'].mean()

avg_price_by_year
```

[118]:
```
year
2001     20000.000000
2004     18000.000000
2005     10000.000000
2006     11500.000000
2007     24983.333333
2008     19871.428571
2009     22642.857143
2010     32057.142857
2011     36787.878788
2012     32810.486486
2013     52349.056604
2014     49453.030303
2015     54550.000000
2016     53597.440476
2017     70529.411765
2018     87837.662338
2019    111125.000000
2020    190000.000000
Name: selling_price, dtype: float64
```

[119]:
```
df['avg_price_by_year'] = df['year'].map(avg_price_by_year)
```

[121]:
```
df.head()
```

[121]:
```
                                name  selling_price  year seller_type  \
2  Royal Enfield Classic Gunmetal Grey         150000  2018  Individual
3    Yamaha Fazer FI V 2.0 [2016-2018]          65000  2015  Individual
5                    Honda CB Twister          18000  2010  Individual
6                Honda CB Hornet 160R          78500  2018  Individual
```

```
9                  Bajaj Discover 125          50000  2016  Individual

       owner  km_driven  ex_showroom_price  Age  avg_price_by_year
2  1st owner      12000           148114.0    7       87837.662338
3  1st owner      23000            89643.0   10       54550.000000
5  1st owner      60000            53857.0   15       32057.142857
6  1st owner      17000            87719.0    7       87837.662338
9  1st owner      42000            60122.0    9       53597.440476
```

[122]:
```
above_avg_price = df[df['selling_price'] > 1.2 * df['avg_price_by_year']]
```

[124]:
```
bikes_above_avg = above_avg_price['name'].unique()
print("Bikes significantly above average price for their year:")
print(bikes_above_avg)
```

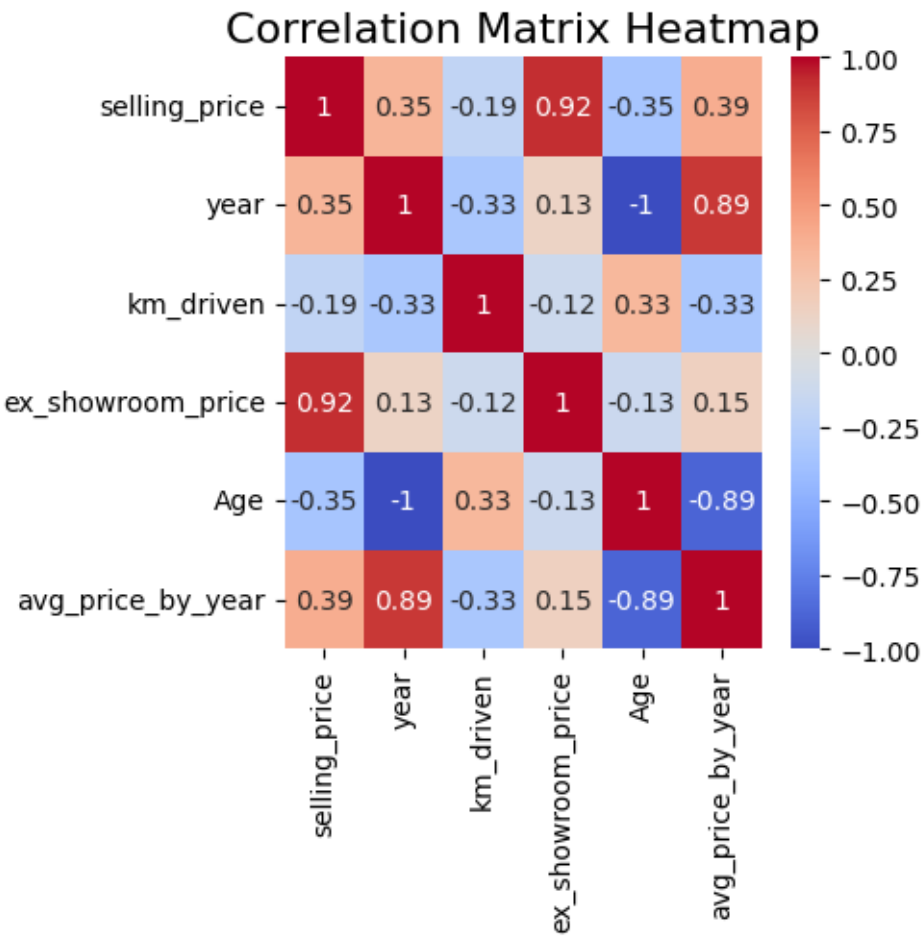
```
Bikes significantly above average price for their year:
['Royal Enfield Classic Gunmetal Grey' 'Yamaha YZF R3' 'Yamaha YZF R15 S'
 'Yamaha FZ25' 'Honda CBR-250R' 'Yamaha FZ16' 'Honda CB Hornet 160R'
 'Mahindra Mojo XT300' 'Royal Enfield Thunderbird 350X'
 'Royal Enfield Classic Desert Storm' 'UM Renegade Commando'
 'Harley-Davidson Street Bob' 'KTM 390 Duke ABS [2013-2016]'
 'Royal Enfield Classic 500' 'Honda CB Unicorn 160' 'Honda CB Unicorn 150'
 'Yamaha YZF R15 [2011-2018]' 'Bajaj Pulsar 180'
 'Royal Enfield Bullet 500' 'Bajaj Dominar 400 [2018]' 'Yamaha FZ V 2.0'
 'Bajaj Avenger Street 220' 'Kawasaki Ninja 650 [2018-2019]'
 'Kawasaki Ninja 250R' 'Suzuki GSX S750' 'Harley-Davidson Street 750'
 'BMW G310GS' 'Royal Enfield Thunderbird 500' 'Hero Achiever 150'
 'Royal Enfield Classic Stealth Black' 'Yamaha Fazer [2009-2016]'
 'Royal Enfield Classic Squadron Blue' 'Royal Enfield Classic Chrome'
 'Royal Enfield Classic Signals' 'Honda CB Unicorn'
 'Bajaj Avenger Street 150 [2018]' 'Yamaha Fazer 25' 'Honda CBR150 R'
 'Bajaj Avenger [2015]' 'Honda Activa [2000-2015]' 'Bajaj Discover 150'
 'Honda CB Unicorn Dazzler' 'Honda Aviator' 'Yamaha FZ S [2012-2016]'
 'Honda CB Twister' 'Yamaha YZF R15 V3' 'Benelli TNT 25'
 'Royal Enfield Thunderbird 500X'
 'Royal Enfield Continental GT [2013 - 2018]' 'Hero Karizma ZMR'
 'Kawasaki Ninja 300' 'TVS Star City' 'Royal Enfield Thunder 500'
 'UM Renegade Mojave' 'KTM RC200' 'Bajaj Dominar 400'
 'Royal Enfield Classic 350' 'KTM RC390' 'Hyosung GT250R'
 'Royal Enfield Thunder 350' 'KTM 390 Duke ' 'Honda CBR 150'
 'TVS Apache RTR 160' 'Bajaj Pulsar 220 F' 'Bajaj Avenger 220 dtsi']
```

Q Develop a correlation matrix for numeric columns and visualize it using a heatmap.

[126]:
```
numeric_column = df.select_dtypes(include = ['number'])
corr_matrix = numeric_column.corr()

#to create a heatmap
```

```
plt.figure(figsize = (4,4))
sns.heatmap(corr_matrix, annot = True, cmap = 'coolwarm')
plt.title('Correlation Matrix Heatmap', fontsize=16)
plt.show()
```



[ ]: