


```
1 from google.colab import files
2 uploaded = files.upload()
```


 Choose Files

Elite Sports... in Data.csv

- Elite Sports Cars in Data.csv(text/csv) - 820452 bytes, last modified: 4/6/2025 - 100% done

Saving Elite Sports Cars in Data.csv to Elite Sports Cars in Data.csv


```
1 import pandas as pd
2
3 # Load CSV
4 df = pd.read_csv("Elite Sports Cars in Data.csv")
5
6 # Show first few rows
7 df.head()
```



	Brand	Model	Year	Country	Condition	Engine_Size	Horsepower	Torque	Weight	Top_Speed	...	Mileage	Popularity	Safety_Ra
0	Nissan	720S	2006	Asia	used	3.7	420	705	1785	238	...	96664	Low	
1	McLaren	911 Turbo S	2009	Europe	new	5.3	1104	766	992	386	...	159630	High	
2	Chevrolet	M4 Competition	2009	USA	new	5.5	153	1573	2022	397	...	111496	High	
3	Bugatti	Chiron	1982	Asia	used	5.4	544	1009	1091	151	...	217228	High	
4	Nissan	Chiron	2022	Europe	new	2.4	980	693	1232	385	...	150318	Low	

5 rows × 27 columns

```
1 #Series Object
2 hp_series = pd.Series(df['Horsepower'])
3 print("Series:\n", hp_series.head())
```

 Series:

0 420

1 1104


2 153

3 544

4 980

Name: Horsepower, dtype: int64

```
1 # Series indexing
2 print("First HP value:", hp_series[0])
3
4 # DataFrame column selection
5 print("Brand column:\n", df['Brand'].head())
6
7 # Row selection by index
8 print("Row at index 2:\n", df.iloc[2])
9
10 # Selecting a range of rows
11 print("Rows 0 to 4:\n", df[0:5])
```



Brand

Chevrolet

Model

M4 Competition

Year

2009

Country

USA

Condition

new

Engine_Size

5.5

Horsepower

153

Torque

1573

Weight

2022

Top_Speed

397

Acceleration_0_100

6.7

Fuel_Type

Diesel

Drivetrain

FWD

```

insurance_cost      1/1b
Production_Units    20000
Log_Price           12.948902
Log_Mileage         11.621753
Modification        NaN
Name: 2, dtype: object
Rows 0 to 4:
   Brand      Model  Year Country Condition  Engine_Size  Horsepower \
0  Nissan      720S  2006   Asia      used          3.7         420
1  McLaren    911 Turbo S  2009  Europe    new          5.3        1104
2  Chevrolet  M4 Competition  2009   USA      new          5.5        153
3  Bugatti     Chiron  1982   Asia      used          5.4        544
4  Nissan      Chiron  2022  Europe    new          2.4        980

   Torque  Weight  Top_Speed  ...  Mileage  Popularity  Safety_Rating \
0     705    1785     238  ...    96664         Low          2
1     766     992     386  ...   159630        High          2
2    1573    2022     397  ...   111496        High          1
3    1009    1091     151  ...   217228        High          2
4     693    1232     385  ...   150318         Low          3

   Number_of_Owners  Market_Demand  Insurance_Cost  Production_Units \
0                 4         Medium        13410          5000
1                 2         Medium        10795          1000
2                 2          Low         1716        20000
3                 4         Medium        11618        20000
4                 2         Medium        11324       100000

   Log_Price  Log_Mileage  Modification
0  11.309352   11.479007         V-Spec
1  12.639334   11.980620          NaN
2  12.948902   11.621753          NaN
3  11.725542   12.288707          NaN
4  11.229289   11.920515          NaN

```

[5 rows x 27 columns]

```

1 #Universal Functions for Index Preservation
2 import numpy as np
3
4 # Apply NumPy ufunc
5 log_hp = np.log1p(df['Horsepower'])
6 print("Log Horsepower:\n", log_hp.head())

```

```

Log Horsepower:
0    6.042633
1    7.007601
2    5.036953
3    6.300786
4    6.888572
Name: Horsepower, dtype: float64

```

```

1 #Index Alignment and Operations Between Series & DataFrames
2 mean_hp = df['Horsepower'].mean()
3 df['HP_Adjusted'] = df['Horsepower'] - mean_hp
4 print(df[['Horsepower', 'HP_Adjusted']].head())
5

```

```

Horsepower  HP_Adjusted
0          420    -402.8916
1         1104     281.1084
2          153   -669.8916
3          544   -278.8916
4          980    157.1084

```

```

1 #Handling Missing Data
2 # Check for missing data
3 print("Missing values:\n", df.isnull().sum())
4
5 # Fill missing data
6 df_filled = df.fillna(0)
7
8 # Drop rows with any missing data
9 df_dropped = df.dropna()

```

```

Missing values:
Brand      0
Model      0
Year       0
Country    0

```

```

Condition          0
Engine_Size        0
Horsepower         0
Torque             0
Weight            0
Top_Speed          0
Acceleration_0_100 0
Fuel_Type          0
Drivetrain         0
Transmission       0
Fuel_Efficiency    0
CO2_Emissions      0
Price             0
Mileage            0
Popularity         0
Safety_Rating      0
Number_of_Owners   0
Market_Demand      0
Insurance_Cost     0
Production_Units   0
Log_Price          0
Log_Mileage        0
Modification       3023
HP_Adjusted        0
dtype: int64

```

```

1 #Operating on Null Values
2 # Detect nulls in Top_Speed
3 null_speed = df['Top_Speed'].isnull()
4
5 # Replace nulls with average value
6 df['Top_Speed'] = df['Top_Speed'].fillna(df['Top_Speed'].mean())

```

```

1 #Hierarchical Indexing (MultiIndex)
2 # Set MultiIndex
3 df_multi = df.set_index(['Brand', 'Model'])
4
5 # Access data using hierarchical index
6 print("Access using MultiIndex:\n", df_multi.loc[('Bugatti', 'Chiron')])

```

→ Access using MultiIndex:

		Year	Country	Condition	Engine_Size	Horsepower	Torque	\
Brand	Model							
Bugatti	Chiron	1982	Asia	used	5.4	544	1009	
	Chiron	1989	USA	new	7.3	561	533	
	Chiron	2024	Europe	new	7.2	893	219	
	Chiron	2021	USA	new	4.1	617	1444	
	Chiron	1987	USA	new	6.6	235	1184	
	Chiron	2008	USA	used	6.4	504	1417	
	Chiron	1987	Asia	salvage	7.7	1432	1736	
	Chiron	2023	Europe	used	4.9	441	682	
	Chiron	1982	Europe	used	5.5	750	1200	
	Chiron	1989	Asia	new	7.6	815	736	
	Chiron	1987	Asia	new	6.3	1129	614	
	Chiron	1990	USA	used	1.8	765	452	
	Chiron	2008	Asia	used	5.6	1086	1339	
	Chiron	1985	Asia	new	6.1	1426	421	
	Chiron	1991	USA	restored	6.3	970	891	
	Chiron	2012	Europe	used	4.7	456	261	
	Chiron	1987	USA	new	2.7	551	964	
	Chiron	1992	Europe	new	7.6	727	1140	
	Chiron	1985	Asia	new	5.3	1447	991	
	Chiron	2006	Europe	new	5.3	817	196	
	Chiron	2022	USA	new	2.5	605	1750	
	Chiron	1993	Europe	used	6.1	212	243	
	Chiron	2012	Asia	used	2.1	734	816	
	Chiron	1984	Asia	new	7.4	766	528	
	Chiron	2019	USA	used	3.8	1355	1328	
	Chiron	2007	Asia	new	1.9	868	927	
	Chiron	1996	Asia	new	4.3	442	211	
	Chiron	1989	Europe	new	6.9	961	1654	
	Chiron	1991	Asia	new	4.0	859	666	
	Chiron	1997	Europe	used	3.8	1386	1157	
	Chiron	1994	USA	salvage	3.6	306	655	
	Chiron	1995	Asia	new	3.6	1143	1508	
	Chiron	1991	Asia	used	4.3	425	579	
	Chiron	1983	Europe	new	4.3	1300	780	
	Chiron	1992	Europe	used	8.0	1452	425	
	Chiron	1981	USA	used	6.0	280	599	
	Chiron	1998	USA	salvage	6.7	329	1099	
	Chiron	1998	Europe	new	6.9	475	1749	

Chiron	1982	Asia	used	2.2	908	1066
Chiron	1995	Asia	new	6.2	1082	1203
Chiron	1992	Asia	used	3.2	1466	902
Chiron	2008	Asia	new	3.8	863	125
Chiron	1996	Europe	new	4.6	471	522
Chiron	2013	Europe	new	5.6	284	127
Chiron	1990	Asia	new	4.7	811	1068
Chiron	2004	Asia	new	2.9	1046	926
Chiron	1988	Asia	new	5.3	550	1664
Chiron	2010	USA	used	5.6	591	847
Chiron	1998	USA	used	4.5	1164	1081
Chiron	1993	USA	restored	6.7	1202	1541
Chiron	1995	Asia	new	6.7	235	1595
Chiron	2015	Europe	new	5.2	179	363
Chiron	1985	Asia	new	3.2	1007	894