My constraints:

- Make the data w.r.t these
  - Price: 10 Lakhs + 50 Thousand (On-Road Price)
    - No need to calculate Ex-Showroom and On-Road Price (set criteria 8 >= 'price_lakhs' <= 10)
  - Mileage >= 13 kmpl
  - Fuel Type: Only Petrol or diesel, consider price.
  - Transmission: Manual or Automatic(Only CVT), consider price.

1. Loading and Viewing data

In [1]:
```
import pandas as pd

cars = pd.read_csv('india_cars_2024.csv', encoding='unicode_escape')

print(f'Dimensionality of the DataFrame:\nRows: {cars.shape[0]}\nColumns: {cars.

cars.head()
```

```
Dimensionality of the DataFrame:
Rows: 1663
Columns: 57
```

Out[1]:

	brand_parent	model_parent	variant_parent	variant_name	price	displacement
0	Volvo	Volvo C40 Recharge	C40 Recharge E80	Volvo C40 Recharge E80	Rs.62.95 Lakh*Get On-Road Price*Ex-showroom Pr...	NaN
1	Volvo	Volvo XC40 Recharge	XC40 Recharge E80 ultimate	Volvo XC40 Recharge E80 ultimate	Rs.57.90 Lakh*Get On-Road Price*Ex-showroom Pr...	NaN
2	Volvo	Volvo XC40 Recharge	XC40 Recharge E60 Plus	Volvo XC40 Recharge E60 Plus	Rs.54.95 Lakh*Get On-Road Price*Ex-showroom Pr...	NaN
3	Volvo	Volvo XC60	XC60 B5 Ultimate	Volvo XC60 B5 Ultimate	Rs.68.90 Lakh*Get On-Road Price*Ex-showroom Pr...	1969 cc
4	Volvo	Volvo S90	S90 B5 Ultimate	Volvo S90 B5 Ultimate	Rs.68.25 Lakh*Get On-Road Price*Ex-showroom Pr...	1969 cc

5 rows × 57 columns

In [2]:

```
print("Number of columns: ", len(cars.columns))
cars.columns
```

```
Number of columns:  57
```

Out[2]:

```
Index(['brand_parent', 'model_parent', 'variant_parent', 'variant_name',
       'price', 'displacement', 'power', 'transmission', 'mileage', 'fuel',
       'engine_typr', 'cylinders', 'valves_per_cyl', 'gearbox', 'drive',
       'fuel_cap', 'emission_norm', 'suspension_front', 'suspension_rear',
       'steer_type', 'steer_col', 'turn_radius', 'brake_front', 'breake_rear',
       'length', 'width', 'height', 'boot_cap', 'seat_cap', 'ground_cl',
       'wheelbase', 'kerb_weight', 'gross_weight', 'doors', 'park_sensors',
       'upholstery', 'radio_antenna', 'tyre_size', 'tyre_type', 'wheel_size',
       'airbags', 'screen_size', 'connectivity', 'image-src', 'ncap_rating',
       'ev_range', 'ev_battery_cap', 'ev_motor', 'ev_charge', 'ev_drag_coeff',
       'zero_to_hundred', 'speakers', 'auto_park', 'ev_charge_time_dc',
       'ev_charge_time_ac', 'ev_brake_regen_levels', 'ev_norm'],
      dtype='object')
```

In [3]:

```
cars.isnull().sum().sort_values(ascending=False).head() # To find if there is an
```

Out[3]:
```
connectivity      1663
ev_motor          1663
image-src         1643
ev_drag_coeff     1634
auto_park         1619
dtype: int64
```

2. Removing unwanted columns

removing columns would help make the data slim as well as faster to process(less dimensions)

Reasons:

- As we're not preferring EVs and CNGs, so dropping all columns related to EVs and CNGs.
  - Not preferring EVs, due to - (A significant hike in electricity charges, if we charge the vehicle at home w.r.t our commute requirements)
  - Not preferring CNG, due to - (Less boot space, Low acceleration performance, Cold start issue, Lower HP, High maintenance cost, Low resale value)
- Connectivity may be referring to infotainment system or App intergration, we could drop it straightaway as it is null for all rows.

In [4]:
```python
ev_cols = [column for column in cars if column.startswith('ev')]
ev_cols
```

Out[4]:
```
['ev_range',
 'ev_battery_cap',
 'ev_motor',
 'ev_charge',
 'ev_drag_coeff',
 'ev_charge_time_dc',
 'ev_charge_time_ac',
 'ev_brake_regen_levels',
 'ev_norm']
```

In [5]:
```python
cars.drop(ev_cols, axis=1, inplace=True)
cars.shape # shape after EV columns dropping, see count of columns
```

Out[5]:
```
(1663, 48)
```

In [6]:
```python
cars['radio_antenna'].unique()
```

Out[6]:
```
array(['Shark Fin', nan, 'shark fin',
       '3-eye Bi-Beam LED headlamps with auto-leveling system And Headlamp Clea
ner, LED turn signal lamps, LED DRL (Daytime Running Lamp)W/o Cut Switch, LED F
ront and Rear fog lamps, LED Rear Combination Lamp & Light Bar Lamp End to End,
Cornering Lamp, LED High Mount Stop lamp (On Rear Spoiler), Panoramic roof (Sli
de UV & IR Cut), Roof Rail(Black), Outside rear view mirror (Auto,EC,Heater)(Vi
sor Cover - Black Paint + IR Function), EMT (Extended Mobility Tire), Front Bum
per & Grille / Rear Bumper(F-Sport), F-Sport front fender emblems, fender arch
moldings, Windshield & Front Side glass - Green UV Acoustic, Front, Rear QTR Gl
ass & Back Glass -Green UV, Rear Side Glass -Light Green UV, Antenna - Radio +S
hark Fin',
       '3-eye Bi-Beam LED headlamps with auto-leveling system And Headlamp Clea
ner, LED turn signal lamps, LED DRL (Daytime Running Lamp) with Cut Switch, LED
Front and Rear fog lamps, LED Rear Combination Lamp & Light Bar Lamp End to End
, Cornering Lamp, LED High Mount Stop lamp (On Rear Spoiler), Panoramic roof (S
lide UV & IR Cut), Roof Rail Silver (Silver), Outside rear view mirror (Auto,E
C,Heater) (Visor cover -Body Color + IR Function), EMT (Extended Mobility Tir
e), Front Bumper & Grille / Rear Bumper ( Normal), Windshield & Front Side glas
s - Green UV Acoustic, Front, Rear QTR Glass & Back Glass -Green UV, Rear Side
Glass -Light Green UV, Antenna - Radio +Shark Fin',
       'Antenna - Radio +Shark Fin, 3-eye Bi-Beam LED headlamps with auto-level
ing system And Headlamp Cleaner, LED turn signal lamps, LED DRL (Daytime Runnin
g Lamp) (W/o Cut Switch), LED Front and Rear fog lamps, LED Rear Combination La
mp & Light Bar Lamp End to End, Cornering Lamp, LED High Mount Stop lamp (On Re
ar Spoiler), Panoramic roof (Slide UV & IR Cut), Roof Rail (Silver), Outside re
ar view mirror (Auto,EC,Heater)(Visor cover- Body Color), EMT (Extended Mobilit
y Tire), Front Bumper & Grille / Rear Bumper (Normal), Windshield & Front Side
glass - Green UV Acoustic, Front, Rear QTR Glass & Back Glass -Green UV, Rear S
ide Glass -Light Green UV,',
       'Dark Grey Metallic Finish Grille,Dark Grey Metallic Finish ORVMs,Body C
olored Door handles,Chrome Tailgate handles,Centre Mounted Roof Antenna,B-pilla
r Black-out Film,Rear Bumper',
       'Shark fin', 'rear Glasss mount antenna', 'shark Fin',
       'Roof Antenna', 'Trail Ready Front Windshield', 'Micro Type',
       'Rear Micro', 'Rod type',
       'Hard Top,All-Black Bumpers,Bonnet Latches,Wheel Arch Cladding,Side Foot
Steps (Moulded),Fender-mounted Radio Antenna,Tailgate mounted Spare Wheel,Illum
inated Key Ring,Body Colour (Satin Matte Desert Fury Colour),ORVMs Inserts (Des
ert Fury Coloured),Vertical slats on the Front grille (Desert Fury Coloured),Ma
hindra Wordmark (Matte Black),Thar branding (Matte Black),4x4 badging (Matte bl
ack With red accents),Automatic badging (Matte black With red accents),Gear Kno
b accents (Dark Chrome)',
       'Fender-mounted', 'Roof antenna', 'Micro', 'Pole (Micro)',
       'SharkFin', 'Micro pole', 'Micro Roof', 'Pole Type', 'Sharkfin',
       'Body Coloured Bumper, Chrome Finish on Rear Bumper, High Mounted LED St
op Lamp, Humanity Line with Chrome Finish, 3-Dimensional Headlamps, Premium Pia
no Black Finish ORVMs, Chrome Lined Door Handles, Fog Lamps with Chrome Ring Su
rrounds, Stylish Finish on B Pillar, Chrome Finish Tri-Arrow Motif Front Grill
e, Chrome Lined Lower Grille, Piano Black Shark Fin Antenna, Sparkling Chrome F
inish Along Window Line, Striking Projector Headlamps',
       'Shark Fin With GPS', 'Glass', 'Micropole'], dtype=object)
```

In [7]:
```
cars.columns
```

Out[7]:
```
Index(['brand_parent', 'model_parent', 'variant_parent', 'variant_name',
       'price', 'displacement', 'power', 'transmission', 'mileage', 'fuel',
       'engine_typr', 'cylinders', 'valves_per_cyl', 'gearbox', 'drive',
       'fuel_cap', 'emission_norm', 'suspension_front', 'suspension_rear',
       'steer_type', 'steer_col', 'turn_radius', 'brake_front', 'breake_rear',
       'length', 'width', 'height', 'boot_cap', 'seat_cap', 'ground_cl',
       'wheelbase', 'kerb_weight', 'gross_weight', 'doors', 'park_sensors',
       'upholstery', 'radio_antenna', 'tyre_size', 'tyre_type', 'wheel_size',
       'airbags', 'screen_size', 'connectivity', 'image-src', 'ncap_rating',
       'zero_to_hundred', 'speakers', 'auto_park'],
      dtype='object')
```

Remove other unwated columns: ['brand_parent', 'model_parent', 'variant_parent', 'radio_antenna', 'gearbox', 'connectivity','image-src', 'engine_typr', 'cylinders', 'valves_per_cyl', 'ncap_rating', 'suspension_front', 'suspension_rear', 'brake_front', 'breake_rear', 'wheelbase', 'kerb_weight','upholstery', 'radio_antenna', 'speakers', 'tyre_size', 'tyre_type', 'wheel_size', 'zero_to_hundred', 'auto_park']

- 'variant_name' is a combination of ('brand_parent', 'model_parent', 'variant_parent')
- 'radio_antenna' is not important to us, and it contain spam info
- All info given by other to-be-deleted-cols will be compared after we get the final names

In [8]:
```
other_unwanted_cols = ['brand_parent', 'model_parent', 'variant_parent', 'radio_

cars.drop(other_unwanted_cols, axis=1, inplace=True)

cars.shape
```

Out[8]: (1663, 24)

For better analysis, we need to change columns with entries like *Value SI unit* to *Value* and change column name to *ColumnName_SI Unit*

Eg: Column 'mileage' has entries '20.36 kmpl', change entry to 20.36 and column name to 'mileage_kmpl'

In [9]:
```
# Extract each values in given column convert it to int, if value is NA then fil

cars['mileage_kmpl'] = cars['mileage'].str.extract(r'(\d+)').fillna(0).astype(in

cars['gross_weight_kg'] = cars['gross_weight'].str.extract(r'(\d+)').fillna(0).a

cars['displacement_cc'] = cars['displacement'].str.extract(r'(\d+)').fillna(0).a

cars['fuel_cap_liters'] = cars['fuel_cap'].str.extract(r'(\d+)').fillna(0).astyp

cars['boot_cap_liters'] = cars['boot_cap'].str.extract(r'(\d+)').fillna(0).astyp


columns_to_process = ['length', 'width', 'height', 'ground_cl']

for column_name in columns_to_process:
```

```
    cars[f'{column_name}_mm'] = cars[column_name].str.extract(r'(\d+)').fillna(0


to_drop_cols = ['mileage', 'gross_weight', 'displacement', 'fuel_cap', 'boot_cap

cars.drop(to_drop_cols, axis=1, inplace=True)

cars.shape
```

Out[9]: (1663, 24)

3. Removing unwanted rows

In [10]:
```
print("Fuels before: ", cars['fuel'].unique()) # Here we Electric, Petrol, Diese

print("No of rows before: ", cars.shape[0])

cars = cars[~cars['fuel'].isin(['Electric', 'CNG'])] # Remove rows where 'fuel'

print("Fuels after: ", cars['fuel'].unique()) # Now we have only cars with 'fuel

print("No of rows after: ", cars.shape[0])

cars.shape
```

```
Fuels before:  ['Electric' 'Petrol' 'Diesel' 'CNG']
No of rows before:  1663
Fuels after:  ['Petrol' 'Diesel']
No of rows after:  1469
```

Out[10]: (1469, 24)

3.1. Fixing the Price Column

Chopping 'Rs.8.07 Lakh x Get On-Road Price x Ex-showroom Price' to 'Rs.8.07 Lakh' for all rows

x is actually asterisk

In [11]:
```
cars['price'] = cars['price'].apply(lambda x: str(x).split('*')[0])
cars['price'].head()
```

Out[11]:
```
3    Rs.68.90 Lakh
4    Rs.68.25 Lakh
5       Rs.1.01 Cr
7       Rs.6.95 Cr
9      Rs.10.48 Cr
Name: price, dtype: object
```

Since we're not thinking of buying cars priced Crores(Cr), so they can be easily removed.

Deleting the rows with price Crores

In [12]:
```
crore_priced_cars = cars['price'].str.contains('cr', case=False, na=False)

print("Total number of rows having 'cr' in price column: ", crore_priced_cars.su
```

```
cars = cars[~crore_priced_cars] # Remove rows where the 'price' column contains

cars.shape
```

```
Total number of rows having 'cr' in price column:  211
```

Out[12]: (1258, 24)

In [13]:
```
pattern = r"rs.|(\s\w+)" # using regex pattern to find multiple strings,
                         # 'rs.' is a word with fullstop,
                         # '\s' finds any whitespace character and '\w+' a compl

# Changing 'Rs.62.95 Lakh' like values to 62.95 (float)
lakh_priced_cars = cars['price'].str \
                                .lower() \
                                .replace(pattern, "", regex=True) \
                                .astype(float)

cars['price_lakhs'] = lakh_priced_cars # Adding new column with name 'price_lakh

cars.drop('price', axis=1, inplace=True) # Dropping the previos 'price' column

print(cars.shape)

cars.head()
```

```
(1258, 24)
```

Out[13]:

	variant_name	power	transmission	fuel	drive	emission_norm	steer_type	steer
3	Volvo XC60 B5 Ultimate	250 bhp	Automatic	Petrol	AWD	BS VI 2.0	Power	T Adjust
4	Volvo S90 B5 Ultimate	246.58 bhp	Automatic	Petrol	FWD	BS VI 2.0	NaN	
36	Porsche Macan Standard	261.49 bhp	Automatic	Petrol	AWD	BS VI 2.0	Power	T Telesc
51	Mini Cooper Countryman Shadow Edition	189.08 bhp	NaN	Petrol	FWD	BS VI 2.0	Power	Adjust Stee
52	Mini Cooper Countryman S JCW Inspired	189.08 bhp	NaN	Petrol	FWD	BS VI 2.0	Power	Adjust Stee

5 rows × 24 columns

4. Comparison with our current car (Honda Amaze 2015)

> The biggest car we can ride through the path to our home is '2024 Maruti Suzuki Brezza' with Dimensions 3,995 mm L x 1,790 mm W x 1,685 mm H

> Amaze has engine displacement 1.2L

> Price range should be in a range of 8-10 Lakhs

> Have mileage greater than 12 kmpl

In [14]:
```
# Conditions combined to filter out rows

cars = cars[
    (cars['price_lakhs'] >= 8.00)      &

    (cars['price_lakhs'] <= 10.00)     &

    (cars['mileage_kmpl'] >= 12)       &

    (cars['length_mm'] <= 3995)        &

    (cars['width_mm'] <= 1790)         &

    (cars['height_mm'] <= 1685)        &

    (cars['boot_cap_liters'] >= 308)
]

cars.shape
```

Out[14]:
```
(112, 24)
```

In [15]:
```
print("Final Cars: ", cars['variant_name'] \
                            .apply(lambda x: ' '.join(x.split()[:2])) \
                            .unique()
)
```

```
Final Cars:  ['Citroen C3' 'Nissan Magnite' 'Renault Kiger' 'Honda Amaze' 'Hyunda
i i20'
 'Hyundai Aura' 'Hyundai Exter' 'Hyundai Venue' 'Toyota Taisor'
 'Tata Tigor' 'Tata Altroz' 'Tata Punch' 'Maruti Dzire' 'Maruti Baleno'
 'Maruti FRONX' 'Maruti Brezza']
```

In [16]:
```
cars.to_csv("required_cars.csv", index=False)
```