Table of Contents

# 1 Static Data Load

import warnings  
warnings.filterwarnings('ignore')

import shap  
print(shap.\_\_version\_\_)

0.48.0

'''  
Created a function 'load\_data\_excel' to load excel into python  
'''  
  
import pandas as pd  
import numpy as np  
import os  
  
def load\_data\_excel(file\_path):  
 """  
 Load data from an Excel file and return a DataFrame.  
 """  
 if not os.path.exists(file\_path):  
 raise FileNotFoundError(f"The file {file\_path} does not exist.")  
   
 df = pd.read\_excel(file\_path)  
 return df

'''  
Loaded VED\_Static\_Data\_ICE&HEV into dataframe df\_ICE\_HEV\_original using the above function  
Loaded VED\_Static\_Data\_PHEV&EV into dataframe df\_ICE\_HEV\_original using the above function  
Created a copy of the original dataframe into df\_ICE\_HEV and df\_PHEV\_EV respectively.  
'''  
  
df\_ICE\_HEV\_original = load\_data\_excel("G:\\DIYguru\\Notes and Sample Data\\VED-master\\Data\\VED\_Static\_Data\_ICE&HEV.xlsx")  
df\_PHEV\_EV\_original = load\_data\_excel("G:\\DIYguru\\Notes and Sample Data\\VED-master\\Data\\VED\_Static\_Data\_PHEV&EV.xlsx")  
  
df\_ICE\_HEV = df\_ICE\_HEV\_original.copy()  
df\_PHEV\_EV = df\_PHEV\_EV\_original.copy()

df\_ICE\_HEV.info()  
df\_PHEV\_EV.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 357 entries, 0 to 356  
Data columns (total 7 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 VehId 357 non-null int64   
 1 Vehicle Type 357 non-null object  
 2 Vehicle Class 357 non-null object  
 3 Engine Configuration & Displacement 357 non-null object  
 4 Transmission 354 non-null object  
 5 Drive Wheels 357 non-null object  
 6 Generalized\_Weight 357 non-null object  
dtypes: int64(1), object(6)  
memory usage: 19.6+ KB  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 27 entries, 0 to 26  
Data columns (total 7 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 VehId 27 non-null int64   
 1 EngineType 27 non-null object  
 2 Vehicle Class 27 non-null object  
 3 Engine Configuration & Displacement 27 non-null object  
 4 Transmission 27 non-null object  
 5 Drive Wheels 27 non-null object  
 6 Generalized\_Weight 27 non-null int64   
dtypes: int64(2), object(5)  
memory usage: 1.6+ KB

'''  
Checked unique values in both the dataframe  
'''  
  
for i in df\_ICE\_HEV.columns:  
 print(f"Unique values in column '{i}':")  
 print(df\_ICE\_HEV[i].unique())

Unique values in column 'VehId':  
[ 2 5 7 8 12 108 110 115 116 119 120 123 124 125 126 128 129 130  
 131 132 133 135 137 138 139 140 141 142 143 145 147 148 149 150 153 154  
 155 156 157 159 160 161 162 163 164 165 167 169 172 174 176 179 180 181  
 184 185 187 189 190 191 192 193 195 196 199 200 201 202 203 205 206 207  
 208 209 211 212 213 214 215 216 217 218 220 222 223 225 228 230 231 232  
 233 234 235 237 238 240 241 242 243 244 246 247 248 249 250 251 252 254  
 255 257 258 259 260 262 264 265 266 267 268 269 270 271 272 273 274 275  
 276 278 280 282 283 285 286 288 289 291 292 293 297 298 299 300 301 302  
 303 304 306 307 308 309 311 312 313 315 318 319 321 323 324 325 326 328  
 329 330 332 333 334 337 338 340 344 345 346 347 348 349 350 351 353 354  
 355 356 357 359 360 366 367 368 369 370 372 374 375 376 378 380 381 382  
 383 384 385 386 387 389 392 393 394 397 399 400 401 402 403 404 405 406  
 407 409 410 411 413 414 415 416 418 422 426 428 429 430 432 433 434 435  
 436 437 438 439 440 441 444 445 447 448 450 451 452 454 456 458 459 460  
 461 462 463 464 465 466 467 468 469 470 472 473 474 475 476 477 478 480  
 482 483 484 485 486 487 488 489 490 494 498 500 501 502 503 504 505 506  
 507 516 517 519 521 522 526 527 528 529 530 531 532 533 534 535 538 539  
 540 543 546 547 548 549 552 555 557 558 562 563 564 565 566 571 573 574  
 575 576 577 578 579 580 581 584 587 588 591 592 595 596 597 598 599 600  
 601 602 603 604 605 606 607 608 609 610 616 618 624 625 630]  
Unique values in column 'Vehicle Type':  
['ICE' 'HEV']  
Unique values in column 'Vehicle Class':  
['Car' 'SUV' 'NO DATA']  
Unique values in column 'Engine Configuration & Displacement':  
['4-FI 2.0L T/C' '4-GAS/ELECTRIC 2.0L' '6-FI 3.6L' '4-FI 1.5L' '4-FI 1.8L'  
 '8-4V/FI 6.0L' '4-GAS/ELECTRIC 2.5L' '10-FI 6.8L' '8-DSL 6.7L T/C'  
 '4-GAS/ELECTRIC 1.8L' '4-GAS/ELECTRIC 2.4L' '8-FI 4.7L' '6-FI 3.4L'  
 '4-FI 2.4L' '8-FI 5.3L ' '6-FI 3.5L' '4-FI 2.5L' '4-FI 2.0L'  
 '4-FI 2.0L PZEV' '8-FI 4.8L' '4-FI 2.2L' '4-FI 1.3L GAS/ELEC.'  
 '5-FI 2.5L' '6-FI 3.7L' '4-FI 1.6L' '4-FI 2.3L T/C' '8-FI 5.4L'  
 '6-FI 4.3L' '5-FI 2.5L PZEV' '4-GAS/ELECTRIC 1.5L' '6-FI 3.3L'  
 '6-FI 3.8L' '4-GAS/ELECTRIC 2.3L' '6-EFI 4.2L ' '6-EFI 3.0L' '8-FI 4.6L'  
 '8-EFI 5.0L' '4-FI T/C 1.4L' '6-FI 3.0L' '6-FI 2.7L' '6-FI 3.1L'  
 '4-FI 1.6L T/C' '8-FI 5.3L' '6-FI 4.2L' '4-FI 2.3L' '4-FI 1.4L T/C'  
 '4-FI 2.3L ULEV' '6-GAS/ELECTRIC 3.3L' '6-GAS/ELECTRIC 3.5L'  
 '3-FI 1.0L GAS/ELEC.' '8-FI 5.7L HEMI (Hemi engine)' '4-FI T/C 2.0L'  
 '8-FI 5.7L HEMI' '6-242-MFI 4.0L' '4-FI S/C 1.8L GAS' '4-FI 1.5L T/C'  
 '4-GAS/ELECTRIC 1.6L' '2.3L Gasoline I4' 'I4 2.4L Flex Fuel' '2.4L'  
 '3.0L 6cyl 4A' 'H-4 2.0 L/122' 'I4 2.2L' 'V6 4.0L' 'V6 3.1L' 'V8 4.7L'  
 'V6 3.0L' 'V6 3.8L' 'V6 3.5L' 'H-4 2.5L' 'I-4 1.8L' 'I-4 2.4L']  
Unique values in column 'Transmission':  
['NO DATA' 'AUTOMATIC' '5-SP MANUAL' '6-SP AUTOMATIC' '5-SP AUTOMATIC'  
 'CVT' 'AUTOMATIC/CVT' '6-SP ECT AUTOMATIC' '5-SP ECT AUTOMATIC'  
 'FULL TIME 4WD AUTOMATIC' '4-SP AUTOMATIC' 'FULL TIME 4WD MANUAL'  
 '5-SP AWD MANUAL' '6-SP AWD MANUAL' '9-SP Automatic' nan '4-SP Automatic'  
 '5-SP Automatic' 'Automatic']  
Unique values in column 'Drive Wheels':  
['NO DATA']  
Unique values in column 'Generalized\_Weight':  
[3500 4500 2500 6000 'NO DATA' 6500 3000 4000 5500 5000 2000]

for i in df\_PHEV\_EV.columns:  
 print(f"Unique values in column '{i}':")  
 print(df\_PHEV\_EV[i].unique())

Unique values in column 'VehId':  
[ 9 10 11 371 379 388 398 417 431 443 449 453 455 457 492 497 536 537  
 541 542 545 550 554 560 561 567 569]  
Unique values in column 'EngineType':  
['PHEV' 'EV']  
Unique values in column 'Vehicle Class':  
['Car']  
Unique values in column 'Engine Configuration & Displacement':  
['4-GAS/ELECTRIC 1.4L' 'ELECTRIC' '4-GAS/ELECTRIC 2.0L'  
 '4-GAS/ELECTRIC 1.8L' '4-GAS/ELECTRIC 1.5L']  
Unique values in column 'Transmission':  
['NO DATA' 'CVT']  
Unique values in column 'Drive Wheels':  
['FWD']  
Unique values in column 'Generalized\_Weight':  
[4000 3500 3000]

'''  
Replaced 'NO DATA' with nan  
'''  
  
df\_ICE\_HEV.replace('NO DATA', np.nan, inplace=True)  
df\_PHEV\_EV.replace('NO DATA', np.nan, inplace=True)

'''  
Handle data type changes and column renames to concatenate two dataframes  
'''  
  
df\_ICE\_HEV['Drive Wheels'] = df\_ICE\_HEV['Drive Wheels'].astype('object')  
df\_PHEV\_EV.rename(columns={'EngineType': 'Vehicle Type'}, inplace=True)

'''  
Created new df\_static by concatenating df\_ICE\_HEV and df\_PHEV\_EV  
'''  
  
df\_static = pd.concat([df\_ICE\_HEV, df\_PHEV\_EV], ignore\_index=True)

df\_static.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 384 entries, 0 to 383  
Data columns (total 7 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 VehId 384 non-null int64   
 1 Vehicle Type 384 non-null object   
 2 Vehicle Class 32 non-null object   
 3 Engine Configuration & Displacement 384 non-null object   
 4 Transmission 95 non-null object   
 5 Drive Wheels 27 non-null object   
 6 Generalized\_Weight 368 non-null float64  
dtypes: float64(1), int64(1), object(5)  
memory usage: 21.1+ KB

'''  
Checked unique and number of unique values in the new df\_static  
'''  
  
for i in df\_static.columns:  
 print(f"\nUnique values in column '{i}':")  
 print(f"Number of unique values: {df\_static[i].nunique()}")  
 print(df\_static[i].unique())  
 print(df\_static[i].value\_counts())

Unique values in column 'VehId':  
Number of unique values: 384  
[ 2 5 7 8 12 108 110 115 116 119 120 123 124 125 126 128 129 130  
 131 132 133 135 137 138 139 140 141 142 143 145 147 148 149 150 153 154  
 155 156 157 159 160 161 162 163 164 165 167 169 172 174 176 179 180 181  
 184 185 187 189 190 191 192 193 195 196 199 200 201 202 203 205 206 207  
 208 209 211 212 213 214 215 216 217 218 220 222 223 225 228 230 231 232  
 233 234 235 237 238 240 241 242 243 244 246 247 248 249 250 251 252 254  
 255 257 258 259 260 262 264 265 266 267 268 269 270 271 272 273 274 275  
 276 278 280 282 283 285 286 288 289 291 292 293 297 298 299 300 301 302  
 303 304 306 307 308 309 311 312 313 315 318 319 321 323 324 325 326 328  
 329 330 332 333 334 337 338 340 344 345 346 347 348 349 350 351 353 354  
 355 356 357 359 360 366 367 368 369 370 372 374 375 376 378 380 381 382  
 383 384 385 386 387 389 392 393 394 397 399 400 401 402 403 404 405 406  
 407 409 410 411 413 414 415 416 418 422 426 428 429 430 432 433 434 435  
 436 437 438 439 440 441 444 445 447 448 450 451 452 454 456 458 459 460  
 461 462 463 464 465 466 467 468 469 470 472 473 474 475 476 477 478 480  
 482 483 484 485 486 487 488 489 490 494 498 500 501 502 503 504 505 506  
 507 516 517 519 521 522 526 527 528 529 530 531 532 533 534 535 538 539  
 540 543 546 547 548 549 552 555 557 558 562 563 564 565 566 571 573 574  
 575 576 577 578 579 580 581 584 587 588 591 592 595 596 597 598 599 600  
 601 602 603 604 605 606 607 608 609 610 616 618 624 625 630 9 10 11  
 371 379 388 398 417 431 443 449 453 455 457 492 497 536 537 541 542 545  
 550 554 560 561 567 569]  
VehId  
449 1  
443 1  
431 1  
417 1  
398 1  
 ..  
12 1  
8 1  
7 1  
5 1  
2 1  
Name: count, Length: 384, dtype: int64  
  
Unique values in column 'Vehicle Type':  
Number of unique values: 4  
['ICE' 'HEV' 'PHEV' 'EV']  
Vehicle Type  
ICE 264  
HEV 93  
PHEV 24  
EV 3  
Name: count, dtype: int64  
  
Unique values in column 'Vehicle Class':  
Number of unique values: 2  
['Car' 'SUV' nan]  
Vehicle Class  
Car 31  
SUV 1  
Name: count, dtype: int64  
  
Unique values in column 'Engine Configuration & Displacement':  
Number of unique values: 74  
['4-FI 2.0L T/C' '4-GAS/ELECTRIC 2.0L' '6-FI 3.6L' '4-FI 1.5L' '4-FI 1.8L'  
 '8-4V/FI 6.0L' '4-GAS/ELECTRIC 2.5L' '10-FI 6.8L' '8-DSL 6.7L T/C'  
 '4-GAS/ELECTRIC 1.8L' '4-GAS/ELECTRIC 2.4L' '8-FI 4.7L' '6-FI 3.4L'  
 '4-FI 2.4L' '8-FI 5.3L ' '6-FI 3.5L' '4-FI 2.5L' '4-FI 2.0L'  
 '4-FI 2.0L PZEV' '8-FI 4.8L' '4-FI 2.2L' '4-FI 1.3L GAS/ELEC.'  
 '5-FI 2.5L' '6-FI 3.7L' '4-FI 1.6L' '4-FI 2.3L T/C' '8-FI 5.4L'  
 '6-FI 4.3L' '5-FI 2.5L PZEV' '4-GAS/ELECTRIC 1.5L' '6-FI 3.3L'  
 '6-FI 3.8L' '4-GAS/ELECTRIC 2.3L' '6-EFI 4.2L ' '6-EFI 3.0L' '8-FI 4.6L'  
 '8-EFI 5.0L' '4-FI T/C 1.4L' '6-FI 3.0L' '6-FI 2.7L' '6-FI 3.1L'  
 '4-FI 1.6L T/C' '8-FI 5.3L' '6-FI 4.2L' '4-FI 2.3L' '4-FI 1.4L T/C'  
 '4-FI 2.3L ULEV' '6-GAS/ELECTRIC 3.3L' '6-GAS/ELECTRIC 3.5L'  
 '3-FI 1.0L GAS/ELEC.' '8-FI 5.7L HEMI (Hemi engine)' '4-FI T/C 2.0L'  
 '8-FI 5.7L HEMI' '6-242-MFI 4.0L' '4-FI S/C 1.8L GAS' '4-FI 1.5L T/C'  
 '4-GAS/ELECTRIC 1.6L' '2.3L Gasoline I4' 'I4 2.4L Flex Fuel' '2.4L'  
 '3.0L 6cyl 4A' 'H-4 2.0 L/122' 'I4 2.2L' 'V6 4.0L' 'V6 3.1L' 'V8 4.7L'  
 'V6 3.0L' 'V6 3.8L' 'V6 3.5L' 'H-4 2.5L' 'I-4 1.8L' 'I-4 2.4L'  
 '4-GAS/ELECTRIC 1.4L' 'ELECTRIC']  
Engine Configuration & Displacement  
4-FI 2.4L 48  
4-GAS/ELECTRIC 1.8L 33  
4-GAS/ELECTRIC 1.5L 32  
4-FI 2.5L 28  
6-FI 3.5L 28  
 ..  
V6 3.0L 1  
H-4 2.5L 1  
V6 3.8L 1  
I-4 2.4L 1  
I-4 1.8L 1  
Name: count, Length: 74, dtype: int64  
  
Unique values in column 'Transmission':  
Number of unique values: 17  
[nan 'AUTOMATIC' '5-SP MANUAL' '6-SP AUTOMATIC' '5-SP AUTOMATIC' 'CVT'  
 'AUTOMATIC/CVT' '6-SP ECT AUTOMATIC' '5-SP ECT AUTOMATIC'  
 'FULL TIME 4WD AUTOMATIC' '4-SP AUTOMATIC' 'FULL TIME 4WD MANUAL'  
 '5-SP AWD MANUAL' '6-SP AWD MANUAL' '9-SP Automatic' '4-SP Automatic'  
 '5-SP Automatic' 'Automatic']  
Transmission  
CVT 27  
5-SP AUTOMATIC 22  
AUTOMATIC/CVT 12  
AUTOMATIC 5  
4-SP AUTOMATIC 5  
4-SP Automatic 5  
5-SP MANUAL 4  
6-SP AUTOMATIC 3  
5-SP Automatic 3  
FULL TIME 4WD AUTOMATIC 2  
6-SP ECT AUTOMATIC 1  
5-SP ECT AUTOMATIC 1  
FULL TIME 4WD MANUAL 1  
6-SP AWD MANUAL 1  
5-SP AWD MANUAL 1  
9-SP Automatic 1  
Automatic 1  
Name: count, dtype: int64  
  
Unique values in column 'Drive Wheels':  
Number of unique values: 1  
[nan 'FWD']  
Drive Wheels  
FWD 27  
Name: count, dtype: int64  
  
Unique values in column 'Generalized\_Weight':  
Number of unique values: 10  
[3500. 4500. 2500. 6000. nan 6500. 3000. 4000. 5500. 5000. 2000.]  
Generalized\_Weight  
3500.0 114  
3000.0 107  
4000.0 53  
2500.0 42  
4500.0 37  
5000.0 6  
5500.0 4  
6000.0 3  
6500.0 1  
2000.0 1  
Name: count, dtype: int64

df\_static.describe(include='all')

VehId Vehicle Type Vehicle Class \  
count 384.000000 384 32   
unique NaN 4 2   
top NaN ICE Car   
freq NaN 264 31   
mean 360.867188 NaN NaN   
std 153.293829 NaN NaN   
min 2.000000 NaN NaN   
25% 236.500000 NaN NaN   
50% 369.500000 NaN NaN   
75% 485.250000 NaN NaN   
max 630.000000 NaN NaN   
  
 Engine Configuration & Displacement Transmission Drive Wheels \  
count 384 95 27   
unique 74 17 1   
top 4-FI 2.4L CVT FWD   
freq 48 27 27   
mean NaN NaN NaN   
std NaN NaN NaN   
min NaN NaN NaN   
25% NaN NaN NaN   
50% NaN NaN NaN   
75% NaN NaN NaN   
max NaN NaN NaN   
  
 Generalized\_Weight   
count 368.000000   
unique NaN   
top NaN   
freq NaN   
mean 3483.695652   
std 697.215509   
min 2000.000000   
25% 3000.000000   
50% 3500.000000   
75% 4000.000000   
max 6500.000000

'''  
Checked duplicates  
'''  
  
duplicates = df\_static.duplicated()  
if duplicates.any():  
 print("Duplicates found in the DataFrame:")  
 print(df\_static[duplicates])

# 2 Dynamic Data Load

'''  
Created a function 'load\_csv\_files\_from\_directory' to load multiple csv files from a directory into python  
'''  
  
def load\_csv\_files\_from\_directory(directory):  
 """  
 Load all CSV files from a specified directory and return a concatenated DataFrame.  
 """  
 all\_files = [f for f in os.listdir(directory) if f.endswith('.csv')]  
 df\_list = []  
   
 for file in all\_files:  
 file\_path = os.path.join(directory, file)  
 df = pd.read\_csv(file\_path)  
 df\_list.append(df)  
   
 return pd.concat(df\_list, ignore\_index=True)

'''  
Loaded VED\_DynamicData\_Part1 into dataframe df\_part1\_original using the above function  
Loaded VED\_DynamicData\_Part2 into dataframe df\_part2\_original using the above function  
Created a copy of the original dataframe into df\_part1 and df\_part2 respectively.  
'''  
  
df\_part1\_original = load\_csv\_files\_from\_directory("G:\\DIYguru\\Notes and Sample Data\\VED-master\\Data\\VED\_DynamicData\_Part1")  
#df\_part2\_original = load\_csv\_files\_from\_directory("G:\\DIYguru\\Notes and Sample Data\\VED-master\\Data\\VED\_DynamicData\_Part2")  
  
df\_part1 = df\_part1\_original.copy()  
#df\_part2 = df\_part2\_original.copy()

df\_part1.info()  
#df\_part2.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 10236957 entries, 0 to 10236956  
Data columns (total 22 columns):  
 # Column Dtype   
--- ------ -----   
 0 DayNum float64  
 1 VehId int64   
 2 Trip int64   
 3 Timestamp(ms) int64   
 4 Latitude[deg] float64  
 5 Longitude[deg] float64  
 6 Vehicle Speed[km/h] float64  
 7 MAF[g/sec] float64  
 8 Engine RPM[RPM] float64  
 9 Absolute Load[%] float64  
 10 OAT[DegC] float64  
 11 Fuel Rate[L/hr] float64  
 12 Air Conditioning Power[kW] float64  
 13 Air Conditioning Power[Watts] float64  
 14 Heater Power[Watts] float64  
 15 HV Battery Current[A] float64  
 16 HV Battery SOC[%] float64  
 17 HV Battery Voltage[V] float64  
 18 Short Term Fuel Trim Bank 1[%] float64  
 19 Short Term Fuel Trim Bank 2[%] float64  
 20 Long Term Fuel Trim Bank 1[%] float64  
 21 Long Term Fuel Trim Bank 2[%] float64  
dtypes: float64(19), int64(3)  
memory usage: 1.7 GB

'''  
Checked unique values in both the dataframe  
'''  
  
for i in df\_part1.columns:  
 print(f"Unique values in column '{i}':")  
 print(df\_part1[i].unique())

Unique values in column 'DayNum':  
[ 1.58665119 1.93178629 1.71977381 ... 154.07965256 154.6622428  
 154.03639969]  
Unique values in column 'VehId':  
[ 8 10 11 124 125 130 133 147 154 155 156 160 165 174 176 184 189 195  
 203 207 209 212 216 220 223 228 230 231 233 237 240 241 242 259 260 265  
 267 272 278 298 299 301 304 319 323 334 340 344 350 351 353 355 356 370  
 374 378 387 388 394 410 411 418 430 433 434 438 439 440 445 449 450 451  
 452 456 460 462 468 477 478 488 497 502 507 516 519 521 528 532 535 537  
 540 547 549 550 557 569 574 575 576 579 584 588 108 110 116 128 140 150  
 157 180 181 185 192 215 225 244 246 248 249 250 257 258 276 291 307 315  
 330 367 369 384 392 393 398 403 415 422 431 432 443 444 453 464 465 480  
 486 494 500 530 531 533 543 546 555 560 564 587 595 601 163 196 205 213  
 217 222 234 266 273 283 289 308 309 324 326 345 346 372 375 382 399 400  
 402 426 441 482 483 538 554 565 592 596 597 606 608 126 132 218 232 243  
 264 268 282 292 337 347 359 368 385 405 428 455 459 463 469 472 473 484  
 489 498 506 529 548 561 566 581 115 153 179 200 269 293 312 338 371 407  
 413 417 501 522 12 139 208 271 454 457 504 562 578 143 149 161 235 254  
 311 383 404 536 563 571 167 251 409 416 436 437 467 603 2 162 349 458  
 476 123 142 164 201 202 466 475 487 526 598 610 5 381 406 542 137 191  
 285 286 303 313 329 386 591 605 609 129 135 252 414 435 447 492 577 211  
 490 148 366 389 187 470 206 138 599 360 379 602 429 607 604 120 159 474  
 503 545 580 172 247 380 558 297 214 357 461 255 275 448 325 348 534 328  
 306 573 321 141 190 9 238 539 600 7 169 567 618 616 376 270 288 274  
 333 145 119 541 527 262 397 199]  
Unique values in column 'Trip':  
[ 706 707 1558 ... 2497 1995 1695]  
Unique values in column 'Timestamp(ms)':  
[ 0 200 1100 ... 4463000 4467100 4472400]  
Unique values in column 'Latitude[deg]':  
[42.27755833 42.27825528 42.2790125 ... 42.23203472 42.25125694  
 42.2483775 ]  
Unique values in column 'Longitude[deg]':  
[-83.69874972 -83.69880306 -83.69890111 ... -83.7150525 -83.71462639  
 -83.75627583]  
Unique values in column 'Vehicle Speed[km/h]':  
[ 40. 45. 47. ... 154. 151. 149.]  
Unique values in column 'MAF[g/sec]':  
[22.12999916 6.1500001 21.44000053 ... 38.80999756 55.7899971  
 49.64999771]  
Unique values in column 'Engine RPM[RPM]':  
[2285. 2744. 1982. ... 6230. 4003. 4230.]  
Unique values in column 'Absolute Load[%]':  
[4.90196075e+01 6.74509811e+01 7.80392151e+01 4.43137245e+01  
 5.09803925e+01 3.56862755e+01 3.21568642e+01 2.74509811e+01  
 2.23529415e+01 3.05882359e+01 3.33333321e+01 1.41176472e+01  
 3.17647057e+01 1.88235302e+01 2.35294113e+01 3.13725491e+01  
 3.41176491e+01 3.72549019e+01 1.49019613e+01 1.84313736e+01  
 2.00000000e+01 5.68627472e+01 5.17647057e+01 7.05882339e+01  
 4.86274529e+01 1.56862745e+01 7.29411774e+01 8.11764755e+01  
 4.07843132e+01 2.27450981e+01 2.31372547e+01 2.39215698e+01  
 2.19607849e+01 2.03921566e+01 2.11764717e+01 3.80392151e+01  
 5.52941170e+01 3.29411774e+01 4.27450981e+01 3.64705887e+01  
 5.92156868e+01 7.45098038e+01 5.21568642e+01 4.74509811e+01  
 1.45098038e+01 1.68627453e+01 4.70588226e+01 2.15686283e+01  
 3.45098038e+01 6.54901962e+01 6.39215698e+01 7.72549057e+01  
 7.68627472e+01 nan 5.13725510e+01 0.00000000e+00  
 5.84313736e+01 6.19607849e+01 5.33333359e+01 5.56862755e+01  
 5.49019623e+01 5.64705887e+01 1.25490198e+01 3.37254906e+01  
 1.52941179e+01 5.76470604e+01 1.01960783e+01 4.78431396e+01  
 6.11764717e+01 4.35294113e+01 6.00000000e+01 6.35294113e+01  
 5.01960793e+01 4.62745094e+01 4.66666679e+01 9.01960754e+00  
 4.31372547e+00 1.37254906e+01 1.05882359e+01 9.80392170e+00  
 4.58823547e+01 5.96078453e+01 5.29411774e+01 4.39215698e+01  
 1.60784321e+01 6.27450981e+01 5.41176491e+01 5.25490189e+01  
 1.17647057e+01 1.21568632e+01 4.47058830e+01 4.31372566e+01  
 4.70588255e+00 4.15686264e+01 1.13725491e+01 4.82352943e+01  
 1.09803925e+01 4.00000000e+01 3.25490189e+01 1.92156868e+01  
 2.54901962e+01 5.09803915e+00 2.86274509e+01 6.78431396e+01  
 6.50980377e+01 6.07843132e+01 3.68627472e+01 3.01960793e+01  
 1.96078434e+01 2.78431377e+01 1.76470585e+01 2.07843132e+01  
 1.33333340e+01 2.43137264e+01 2.62745094e+01 5.80392151e+01  
 3.49019623e+01 5.60784340e+01 3.92156868e+01 8.23529434e+00  
 3.09803925e+01 9.41176510e+00 2.82352943e+01 4.11764717e+01  
 2.66666679e+01 2.47058830e+01 2.50980396e+01 6.47058868e+01  
 1.72549019e+01 5.72549019e+01 4.94117661e+01 2.58823528e+01  
 4.23529434e+01 1.64705887e+01 6.43137283e+01 2.90196075e+01  
 1.80392151e+01 5.37254906e+01 6.23529434e+01 6.98039246e+01  
 7.17647095e+01 3.96078453e+01 6.58823547e+01 7.76470566e+01  
 3.52941170e+01 6.90196075e+01 7.92156906e+01 3.60784302e+01  
 4.03921585e+01 2.98039227e+01 1.29411764e+01 3.76470604e+01  
 8.70588226e+01 2.94117661e+01 7.09803925e+01 6.03921585e+01  
 2.70588245e+01 3.88235283e+01 4.54901962e+01 4.98039207e+01  
 6.82352982e+01 7.52941208e+01 8.54901962e+01 5.88235321e+01  
 8.35294113e+01 7.64705887e+01 4.50980415e+01 6.70588226e+01  
 7.37254944e+01 7.21568604e+01 6.86274490e+01 7.84313736e+01  
 8.00000000e+01 5.45098038e+01 6.62745132e+01 8.58823547e+01  
 7.13725510e+01 8.62745094e+00 6.66666641e+01 4.19607849e+01  
 3.84313736e+01 5.05882378e+01 6.15686302e+01 7.41176453e+01  
 6.94117661e+01 7.60784302e+01 6.31372566e+01 5.88235283e+00  
 1.01176476e+02 8.78431396e+01 1.28627457e+02 1.31764709e+02  
 8.15686264e+01 9.09803925e+01 1.05882355e+02 3.92156863e+00  
 7.84313726e+00 5.49019623e+00 7.33333359e+01 7.01960831e+01  
 8.50980377e+01 6.27450991e+00 6.66666698e+00 1.35294113e+02  
 7.88235321e+01 8.39215698e+01 8.43137283e+01 7.96078415e+01  
 8.86274490e+01 9.17647095e+01 8.27451019e+01 8.62745132e+01  
 1.09803925e+02 1.06274513e+02 1.00000000e+02 1.21568626e+02  
 8.98039246e+01 9.25490189e+01 1.09411766e+02 8.82352982e+01  
 7.49019623e+01 9.88235321e+01 7.56862793e+01 7.25490189e+01  
 9.52941208e+01 1.07058823e+02 1.11764709e+02 8.90196075e+01  
 9.33333359e+01 9.76470642e+01 2.35294127e+00 8.23529434e+01  
 7.45098066e+00 9.68627472e+01 8.19607849e+01 9.84313736e+01  
 7.05882359e+00 3.13725495e+00 1.04705887e+02 1.27843140e+02  
 9.01960831e+01 1.00392159e+02 1.78431381e+02 1.27058823e+02  
 9.60784302e+01 9.13725510e+01 1.01960785e+02 1.23921570e+02  
 9.72549057e+01 8.47058868e+01 1.14117645e+02 1.08235298e+02  
 1.01568626e+02 8.07843170e+01 8.03921585e+01 1.24313728e+02  
 8.94117661e+01 8.66666718e+01 1.06666672e+02 9.21568680e+01  
 1.21176476e+02 1.25882355e+02 9.29411774e+01 9.05882339e+01  
 3.52941179e+00 9.37254944e+01 8.74509811e+01 9.64705887e+01  
 1.02352943e+02 8.31372528e+01 1.00784317e+02 1.29019608e+02  
 1.03529411e+02 1.16078430e+02 1.12941177e+02 1.54901962e+02  
 1.17647064e+02 9.45098038e+01 1.34901962e+02 1.35686279e+02  
 1.18823532e+02 1.39607849e+02 1.40784317e+02 1.45098038e+02  
 1.20000000e+02 9.56862793e+01 9.92156906e+01 1.15294121e+02  
 9.41176453e+01 9.96078415e+01 1.25490196e+02 1.09019608e+02  
 1.19215691e+02 1.41176468e+02 9.49019623e+01 2.74509811e+00  
 1.22745102e+02 1.03137253e+02 1.36078430e+02 1.23137260e+02  
 1.31372559e+02 1.12549019e+02 1.93333344e+02 1.14509804e+02  
 1.05098038e+02 1.45490204e+02 1.25098038e+02 1.30196075e+02  
 1.62352951e+02 1.07843140e+02 1.36862747e+02 1.19607841e+02  
 1.26666672e+02 1.11372551e+02 1.56862747e+02 5.43921570e+02  
 1.94117645e+02 9.80392151e+01 1.03921570e+02 1.02745102e+02  
 1.31431377e+04 1.44705887e+02 1.58823532e+02 1.12156868e+02  
 1.27450981e+02 1.36470596e+02 1.42352951e+02 1.26274513e+02  
 1.24705887e+02 1.04313728e+02 1.16862747e+02 1.14901962e+02  
 1.39215683e+02 1.41568634e+02 1.33333328e+02 1.10980392e+02  
 1.16470589e+02 1.32156860e+02 1.96078432e+00 1.55686279e+02  
 1.32941177e+02 1.10588234e+02 1.58039215e+02 1.30588242e+02  
 2.30588242e+02 1.05490196e+02 1.73435293e+04 1.86274506e+02  
 1.66274506e+02 1.52549026e+02 1.08627449e+02 1.18431374e+02  
 1.75686279e+02 1.65639219e+04 1.84313736e+02 1.34117645e+02  
 1.20784317e+02 1.07450981e+02 1.23529411e+02 1.69019608e+02  
 1.10196083e+02 1.49411774e+02 1.50196075e+02 1.56470596e+02  
 1.18039215e+02 1.49019608e+02 1.38823532e+02 1.34509811e+02  
 1.53725494e+02 1.38039215e+02 1.68235291e+02 1.45882355e+02  
 1.64705887e+02 1.77647064e+02 1.90980392e+02 1.22352943e+02  
 1.65098038e+02 5.21568665e+02 1.53333328e+02 1.17254906e+02  
 1.80784317e+02 1.13333336e+02 1.13725494e+02 1.43529419e+02  
 1.38431381e+02 1.70588242e+02 1.57647064e+02 1.74901962e+02  
 1.44313721e+02 1.65882355e+02 1.29803925e+02 2.16470596e+02  
 1.61176468e+02 1.54509811e+02 1.40392166e+02 1.21960785e+02  
 1.33725494e+02 1.51372559e+02 1.28235291e+02 1.63529419e+02  
 1.46666672e+02 1.47843140e+02 1.49803925e+02 1.30980392e+02  
 1.32549026e+02 1.52941177e+02 2.03529419e+02 1.59607849e+02  
 1.60392166e+02 1.46274506e+02 1.95294128e+02 1.37647064e+02  
 1.29411774e+02 2.04313736e+02 1.76862747e+02 1.67450989e+02  
 1.15686279e+02 1.40000000e+02 2.41568634e+02 1.20392159e+02  
 1.13450986e+04 2.13290195e+04 2.06482363e+04 1.29333340e+04  
 1.55294113e+02 1.66666672e+02 1.81568634e+02 2.06666672e+02  
 1.33874512e+04 1.92403926e+04 1.58431381e+02 1.78039215e+02  
 1.91764709e+02 1.69803925e+02 1.36141182e+04 1.70980392e+02  
 1.60345098e+04 1.48235291e+02 1.43921570e+02 1.63137253e+02  
 1.28576475e+04 1.72156860e+02 1.17647064e+00 1.43137253e+02  
 1.89803925e+02 1.74117645e+02 1.59215683e+02 1.48627457e+02  
 1.52156860e+02 1.74509811e+02 1.41960785e+02 1.69411774e+02  
 1.56862748e+00 1.73333344e+02 1.90196075e+02 1.54117645e+02  
 1.47450989e+02 1.47058823e+02 1.45972549e+04 9.37843164e+03  
 1.88329414e+04 1.68666660e+04 1.24039219e+04 1.32360791e+04  
 2.06274506e+02 1.85882355e+02 1.88627457e+02 1.42745102e+02]  
Unique values in column 'OAT[DegC]':  
[ nan 5. 6.25 5.25 5.75 3.75 4. 4.25 4.5 3.5  
 3.25 4.75 5.5 17. 18. 19. 8. 7.5 7. 6.5  
 2.25 2.75 3. 8.75 6.75 7.75 9. 8.25 8.5 9.25  
 10.5 10.75 9.5 9.75 7.25 6. 1.5 2.5 1.75 2.  
 10. 14. 14.5 14.75 11.5 16. 20. 22. 23. 24.  
 25. 26. 27. 28. 29. 30. 15. 12.5 12.25 11.75  
 12.75 13. 10.25 11.25 11. 15.5 13.75 13.5 15.25 16.5  
 12. 21. -3. -2. -1. 0.5 0. 1. -0.5 0.75  
 -0.25 -1.75 1.25 0.25 -1.5 -2.5 -2.75 -2.25 -3.25 -5.  
 -6.25 -5.75 -5.5 -6. -4. -4.5 -3.5 -0.75 -3.75 -5.25  
 -6.5 -4.75 -4.25 -1.25 17.5 18.25 18.5 18.75 -8. -7.  
 -7.25 -8.25 -8.5 -7.5 -7.75 -6.75 -9. -8.75 -9.75 -10.5  
 -10. -9.25 -9.5 -13. -13.5 -12.75 -12.25 -11.5 -11.25 -11.  
 -10.75 -12. -14. -12.5 -10.25 -11.75 -16. -17. -17.5 -17.75  
 -17.25 -18.5 -18. -15. -14.25 -13.75 -14.5 -16.5 -15.5 -15.75  
 -14.75 -15.25 -16.25 -16.75 -13.25 -19. -19.5 -21.25 -24.5 -20.25  
 -20. -19.75 -21.5 -20.5 -21. -18.25 -20.75 32. 39. 35.  
 15.75 16.75 -40. 16.25 33. 14.25 17.75 -39. -38. -37.  
 -36. 34. 13.25]  
Unique values in column 'Fuel Rate[L/hr]':  
[ nan 0. 6.69228172 ... 4.21666241 8.41424465 0.92537612]  
Unique values in column 'Air Conditioning Power[kW]':  
[ nan 5.79999971 5.92000008 5.71999979 5.75999975 5.83999968  
 5.67999983 0.59999996 0. 0.16 0.19999999 0.31999999  
 0.08 0.47999999 0.28 0.51999998 5.5999999 5.55999994  
 5.63999987 0.23999999 0.56 6.07999992 6.19999981 6.15999985  
 0.04 0.39999998 0.35999998 0.44 6.23999977 6.31999969  
 6.03999996 6.35999966 6.27999973 6.11999989 6. 5.96000004  
 5.87999964 0.12 5.27999973 4.75999975 4.19999981 4.83999968  
 5.31999969 5.19999981 5.35999966 5.44000006 5.4000001 5.51999998  
 5.48000002 5.23999977 5.15999985]  
Unique values in column 'Air Conditioning Power[Watts]':  
[ nan 0. 100. 1000. 1350. 240. 200. 280. 560. 640. 400. 360.  
 440. 480. 120. 150. 550. 800. 750. 700. 650. 250. 300. 350.  
 850. 900. 950. 160. 320. 450. 1050. 500. 1100. 1950. 1900. 1850.  
 1800. 1150. 80. 1250. 1200. 1300. 600. 2500. 2050. 2350. 1700. 1750.  
 520. 40. 50. 960. 1400. 680. 2000. 1450. 2300. 2250. 1500.]  
Unique values in column 'Heater Power[Watts]':  
[ nan 2250. 2000. 1750. 1500. 1250. 1000. 750. 500. 250. 0. 3000.  
 3250. 2750. 3500. 3750. 4000. 2500. 4250.]  
Unique values in column 'HV Battery Current[A]':  
[ nan -21.5 23.5 ... -82.69000244 -166.75  
 -160.3999939 ]  
Unique values in column 'HV Battery SOC[%]':  
[ nan 96.34146881 95.97561646 ... 85.94937134 86.73416901  
 83.01265717]  
Unique values in column 'HV Battery Voltage[V]':  
[ nan 386. 390.5 ... 306.875 395.875 394.875]  
Unique values in column 'Short Term Fuel Trim Bank 1[%]':  
[ nan -3.90625 -3.125 9.375 -0.78125 -7.03125 1.5625  
 3.125 0. -1.5625 3.90625 -4.6875 0.78125 -6.25  
 -7.8125 4.6875 6.25 -9.375 7.8125 5.46875 -5.46875  
 10.15625 2.34375 -2.34375 18.75 -22.65625 -16.40625 12.5  
 -23.4375 10.9375 7.03125 -10.9375 8.59375 -8.59375 -11.71875  
 -10.15625 11.71875 36.71875 14.0625 -17.1875 13.28125 -14.0625  
 -17.96875 -15.625 -13.28125 -12.5 -14.84375 -21.09375 -19.53125  
 -20.3125 -18.75 15.625 14.84375 34.375 25. 20.3125  
 23.4375 29.6875 17.96875 -25.78125 -27.34375 33.59375 19.53125  
 21.875 21.09375 28.90625 24.21875 28.125 25.78125 17.1875  
 22.65625 32.8125 16.40625 31.25 38.28125 35.9375 35.15625  
 30.46875 32.03125 40.625 27.34375 26.5625 37.5 39.84375  
 46.09375 39.0625 46.875 44.53125 42.96875 41.40625 43.75  
 -24.21875 47.65625 -29.6875 42.1875 -21.875 -28.90625 -25.  
 -26.5625 45.3125 50.78125 73.4375 88.28125 49.21875 57.8125  
 50. -35.15625 72.65625 -28.125 78.90625 65.625 81.25  
 -33.59375 76.5625 53.125 89.84375 -32.03125 -34.375 52.34375  
 53.90625 51.5625 48.4375 -31.25 ]  
Unique values in column 'Short Term Fuel Trim Bank 2[%]':  
[ nan -3.125 3.125 8.59375 -0.78125 0.78125 6.25  
 7.8125 5.46875 0. -1.5625 -3.90625 -4.6875 -5.46875  
 -7.8125 2.34375 10.9375 -7.03125 1.5625 -2.34375 3.90625  
 4.6875 -6.25 7.03125 10.15625 9.375 -12.5 -8.59375  
 -13.28125 -10.9375 -16.40625 -14.0625 -10.15625 -9.375 -18.75  
 -20.3125 -21.09375 -14.84375 -19.53125 -17.1875 -17.96875 11.71875  
 29.6875 34.375 -11.71875 16.40625 -15.625 17.96875 21.875  
 15.625 21.09375 12.5 14.84375 19.53125 27.34375 17.1875  
 24.21875 32.8125 31.25 14.0625 13.28125 28.125 28.90625  
 30.46875 39.0625 37.5 26.5625 18.75 32.03125 22.65625  
 25. 20.3125 36.71875 25.78125 35.15625 39.84375 33.59375  
 23.4375 35.9375 -21.875 -23.4375 -26.5625 -93.75 -28.90625  
 38.28125 41.40625 40.625 42.1875 45.3125 50.78125 50.  
 -25. -27.34375 -22.65625 -30.46875 -29.6875 -24.21875 52.34375  
 47.65625 46.875 53.125 53.90625 51.5625 42.96875 44.53125  
 43.75 49.21875 46.09375 -25.78125]  
Unique values in column 'Long Term Fuel Trim Bank 1[%]':  
[ nan -3.125 -5.46875 -2.34375 -1.5625 2.34375 1.5625  
 3.125 -0.78125 0. 0.78125 5.46875 6.25 7.03125  
 7.8125 8.59375 9.375 10.15625 -7.03125 -6.25 -7.8125  
 -10.15625 -3.90625 -14.84375 -4.6875 -10.9375 4.6875 3.90625  
 10.9375 -9.375 -8.59375 -13.28125 11.71875 13.28125 12.5  
 -12.5 14.0625 17.1875 16.40625 15.625 14.84375 19.53125  
 21.09375 -14.0625 17.96875 18.75 23.4375 21.875 22.65625  
 24.21875 25. 20.3125 -11.71875 -23.4375 -32.8125 -29.6875  
 -24.21875 -25.78125 -22.65625 -15.625 -20.3125 -26.5625 -21.09375  
 -17.1875 -17.96875 -16.40625 -18.75 -19.53125 -21.875 -27.34375  
 -28.125 -30.46875 -25. 25.78125 35.15625 26.5625 28.90625  
 38.28125 33.59375 28.125 31.25 32.03125 35.9375 42.96875  
 27.34375 37.5 36.71875 39.0625 45.3125 30.46875 43.75  
 39.84375 44.53125 40.625 29.6875 34.375 32.8125 ]  
Unique values in column 'Long Term Fuel Trim Bank 2[%]':  
[ nan -2.34375 -7.03125 -3.125 5.46875 4.6875 7.8125  
 6.25 7.03125 8.59375 -1.5625 -3.90625 -6.25 -5.46875  
 -0.78125 -9.375 -10.15625 -4.6875 -13.28125 -10.9375 -7.8125  
 10.15625 3.125 3.90625 1.5625 2.34375 9.375 0.  
 -12.5 0.78125 21.875 21.09375 17.1875 12.5 10.9375  
 25. 14.84375 11.71875 23.4375 19.53125 13.28125 22.65625  
 15.625 -8.59375 14.0625 -11.71875 17.96875 16.40625 18.75  
 -17.96875 -25. -25.78125 -21.09375 -17.1875 -19.53125 -14.0625  
 -16.40625 -14.84375 24.21875 -20.3125 20.3125 -15.625 -22.65625  
 -30.46875 -18.75 -24.21875 -26.5625 -21.875 -27.34375]

'''  
for i in df\_part2.columns:  
 print(f"Unique values in column '{i}':")  
 print(df\_part2[i].unique())  
  
'''

'''  
Created new df\_dynamic by concatenating df\_part1 and df\_part2  
'''  
  
#df\_dynamic = pd.concat([df\_part1, df\_part2], ignore\_index=True)

#df\_dynamic.info()

'''  
Checked unique and number of unique values in the new df\_dynamic  
  
  
for i in df\_dynamic.columns:  
 print(f"\nUnique values in column '{i}':")  
 print(f"Number of unique values: {df\_dynamic[i].nunique()}")  
 print(df\_dynamic[i].unique())  
 print(df\_dynamic[i].value\_counts())  
  
'''

#df\_dynamic.head()

'''  
Checked duplicates  
  
duplicates = df\_dynamic.duplicated()  
if duplicates.any():  
 print("Duplicates found in the DataFrame:")  
 print(df\_dynamic[duplicates])  
'''

df\_static.columns

#df\_dynamic.columns

# 3 Join Static and Dynamic dataframes

'''  
Dynamic has 22436808 records and so was not able to join because of RAM size  
'''  
  
#print(df\_dynamic['VehId'].nunique(), len(df\_dynamic))  
print(df\_static['VehId'].nunique(), len(df\_static))

384 384

'''  
Even the sample of 50% df\_dynamic didnt work, so pulled 50% sample from df\_part1 instead of df\_dynamic which has 5118478 records  
'''  
  
df\_dynamic\_sample = df\_part1.sample(frac=0.50, random\_state=42)

len(df\_dynamic\_sample)

5118478

'''  
Joined dataframe df with 5118478 entries is created.  
'''  
  
df = df\_dynamic\_sample.merge(df\_static, on='VehId', how='left')

#22436808  
df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5118478 entries, 0 to 5118477  
Data columns (total 28 columns):  
 # Column Dtype   
--- ------ -----   
 0 DayNum float64  
 1 VehId int64   
 2 Trip int64   
 3 Timestamp(ms) int64   
 4 Latitude[deg] float64  
 5 Longitude[deg] float64  
 6 Vehicle Speed[km/h] float64  
 7 MAF[g/sec] float64  
 8 Engine RPM[RPM] float64  
 9 Absolute Load[%] float64  
 10 OAT[DegC] float64  
 11 Fuel Rate[L/hr] float64  
 12 Air Conditioning Power[kW] float64  
 13 Air Conditioning Power[Watts] float64  
 14 Heater Power[Watts] float64  
 15 HV Battery Current[A] float64  
 16 HV Battery SOC[%] float64  
 17 HV Battery Voltage[V] float64  
 18 Short Term Fuel Trim Bank 1[%] float64  
 19 Short Term Fuel Trim Bank 2[%] float64  
 20 Long Term Fuel Trim Bank 1[%] float64  
 21 Long Term Fuel Trim Bank 2[%] float64  
 22 Vehicle Type object   
 23 Vehicle Class object   
 24 Engine Configuration & Displacement object   
 25 Transmission object   
 26 Drive Wheels object   
 27 Generalized\_Weight float64  
dtypes: float64(20), int64(3), object(5)  
memory usage: 1.1+ GB

'''  
Checked for duplicates  
'''  
  
duplicates = df.duplicated()  
if duplicates.any():  
 print("Duplicates found in the DataFrame:")  
 print(df[duplicates])

'''  
Checked unique and number of unique values in the new df\_dynamic  
'''  
  
for i in df.columns:  
 print(f"\nUnique values in column '{i}':")  
 print(f"Number of unique values: {df[i].nunique()}")  
 print(df[i].unique())  
 print(df[i].value\_counts())

Unique values in column 'DayNum':  
Number of unique values: 15383  
[130.55213663 109.90206494 46.50061109 ... 30.59042635 7.60616594  
 76.84865423]  
DayNum  
47.605510 5395  
42.708322 4880  
100.445583 3220  
1.477313 3179  
23.953190 2932  
 ...   
35.892528 42  
141.737495 42  
37.589300 41  
36.048668 40  
114.777630 39  
Name: count, Length: 15383, dtype: int64  
  
Unique values in column 'VehId':  
Number of unique values: 368  
[487 392 285 566 242 560 494 240 369 542 565 428 156 212 455 394 452 443  
 308 465 176 418 237 371 278 577 10 545 301 271 192 289 460 531 562 557  
 181 230 370 208 184 323 272 356 298 459 536 160 283 477 484 534 139 533  
 120 232 569 411 470 195 2 250 180 8 258 540 591 454 319 488 409 415  
 309 564 457 349 374 516 595 135 282 233 185 378 368 223 437 274 140 213  
 483 618 269 228 561 353 372 388 333 163 203 522 299 205 592 430 439 490  
 435 276 530 196 286 447 155 476 266 147 475 347 410 444 215 575 218 351  
 546 468 307 450 603 500 399 251 167 601 596 526 344 579 174 453 458 220  
 359 549 440 306 597 350 201 521 311 11 382 528 489 259 404 337 132 451  
 429 355 143 400 267 115 472 462 142 464 449 422 554 456 393 273 366 469  
 257 116 384 324 265 268 563 550 154 486 231 124 398 249 609 304 5 214  
 537 202 578 529 402 506 441 416 291 507 189 334 433 312 604 329 385 123  
 149 467 244 326 417 602 535 571 478 584 438 607 153 448 381 128 497 599  
 293 547 367 200 548 574 386 133 164 466 340 387 125 252 235 110 260 504  
 150 436 576 248 161 403 234 588 480 558 246 405 463 581 338 187 264 207  
 12 130 216 321 179 445 502 473 407 608 165 492 413 141 328 383 587 346  
 414 431 169 126 292 241 275 137 157 138 503 357 315 206 225 376 243 406  
 434 288 474 191 389 162 543 580 209 519 330 482 555 148 211 348 345 539  
 573 375 254 270 598 538 217 313 501 532 360 567 238 108 129 379 9 297  
 606 605 541 262 325 172 247 303 7 610 498 600 426 159 380 397 199 255  
 527 145 190 222 616 432 119 461]  
VehId  
560 199115  
371 73843  
564 69803  
388 68819  
484 64062  
 ...   
247 268  
145 180  
606 125  
119 121  
461 85  
Name: count, Length: 368, dtype: int64  
  
Unique values in column 'Trip':  
Number of unique values: 2386  
[1168 637 1387 ... 2017 2187 2172]  
Trip  
988 11402  
956 10766  
1035 8823  
1249 8704  
884 7936  
 ...   
35 50  
2017 49  
2355 47  
2041 45  
2549 42  
Name: count, Length: 2386, dtype: int64  
  
Unique values in column 'Timestamp(ms)':  
Number of unique values: 54557  
[ 35200 343300 1287100 ... 6608200 3464300 6701800]  
Timestamp(ms)  
0 7666  
3000 2819  
16000 2641  
3100 2292  
13000 2241  
 ...   
7279800 1  
3466200 1  
6661400 1  
8115300 1  
3930700 1  
Name: count, Length: 54557, dtype: int64  
  
Unique values in column 'Latitude[deg]':  
Number of unique values: 313477  
[42.25612 42.29448944 42.28174472 ... 42.31123833 42.24938417  
 42.31985361]  
Latitude[deg]  
42.281242 750  
42.266517 583  
42.266613 528  
42.275244 517  
42.265952 471  
 ...   
42.242824 1  
42.314930 1  
42.296204 1  
42.231457 1  
42.241955 1  
Name: count, Length: 313477, dtype: int64  
  
Unique values in column 'Longitude[deg]':  
Number of unique values: 410769  
[-83.69247694 -83.68260694 -83.75682806 ... -83.77265056 -83.77151889  
 -83.78413639]  
Longitude[deg]  
-83.739434 723  
-83.747713 585  
-83.744237 536  
-83.747760 526  
-83.751339 465  
 ...   
-83.802023 1  
-83.779499 1  
-83.778095 1  
-83.687196 1  
-83.798014 1  
Name: count, Length: 410769, dtype: int64  
  
Unique values in column 'Vehicle Speed[km/h]':  
Number of unique values: 18816  
[ 58. 63. 49. ... 133.4375 82.4375  
 89.62999725]  
Vehicle Speed[km/h]  
0.000000 641212  
54.000000 79956  
56.000000 78224  
57.000000 77647  
52.000000 76073  
 ...   
82.437500 1  
133.437500 1  
102.369995 1  
133.906250 1  
100.049995 1  
Name: count, Length: 18816, dtype: int64  
  
Unique values in column 'MAF[g/sec]':  
Number of unique values: 9240  
[ 5.53999996 13.39999962 17.52000046 ... 77.87999725 65.26000214  
 65.15000153]  
MAF[g/sec]  
0.710000 162706  
0.170000 79402  
0.870000 67227  
0.180000 56604  
0.180000 41633  
 ...   
65.970001 1  
80.190002 1  
92.059998 1  
79.559998 1  
69.190002 1  
Name: count, Length: 9240, dtype: int64  
  
Unique values in column 'Engine RPM[RPM]':  
Number of unique values: 4432  
[1504. 1135. 1665. ... 3940. 4330. 4611.]  
Engine RPM[RPM]  
0.0 725968  
1280.0 27526  
1184.0 20684  
1312.0 20503  
1248.0 17883  
 ...   
4799.0 1  
4623.0 1  
4716.0 1  
4243.0 1  
3882.0 1  
Name: count, Length: 4432, dtype: int64  
  
Unique values in column 'Absolute Load[%]':  
Number of unique values: 463  
[ nan 3.52941170e+01 4.47058830e+01 0.00000000e+00  
 5.64705887e+01 1.45098038e+01 5.76470604e+01 3.17647057e+01  
 3.09803925e+01 3.25490189e+01 1.52941179e+01 1.33333340e+01  
 1.29411764e+01 4.00000000e+01 5.84313736e+01 2.07843132e+01  
 2.54901962e+01 2.90196075e+01 5.33333359e+01 3.05882359e+01  
 2.58823528e+01 5.01960793e+01 5.37254906e+01 1.72549019e+01  
 8.43137283e+01 2.70588245e+01 3.45098038e+01 1.37254906e+01  
 4.74509811e+01 6.27450981e+01 6.11764717e+01 4.50980415e+01  
 3.84313736e+01 1.25490198e+01 3.92156868e+01 1.41176472e+01  
 2.11764717e+01 5.96078453e+01 1.17647057e+01 2.78431377e+01  
 1.49019613e+01 4.03921585e+01 8.00000000e+01 2.19607849e+01  
 7.13725510e+01 4.94117661e+01 4.66666679e+01 3.49019623e+01  
 3.21568642e+01 1.21568632e+01 4.86274529e+01 2.23529415e+01  
 1.28627457e+02 4.54901962e+01 3.37254906e+01 1.96078434e+01  
 2.31372547e+01 2.39215698e+01 2.98039227e+01 4.27450981e+01  
 1.56862745e+01 5.56862755e+01 4.78431396e+01 1.01960783e+01  
 1.84313736e+01 2.35294113e+01 2.15686283e+01 1.80392151e+01  
 1.29333340e+04 4.90196075e+01 6.47058868e+01 1.60784321e+01  
 5.29411774e+01 2.50980396e+01 1.88235302e+01 5.13725510e+01  
 1.05882359e+01 3.72549019e+01 3.60784302e+01 4.31372547e+00  
 2.00000000e+01 1.92156868e+01 2.82352943e+01 4.98039207e+01  
 9.01960754e+00 3.13725495e+00 5.52941170e+01 1.64705887e+01  
 5.21568642e+01 2.94117661e+01 4.35294113e+01 2.47058830e+01  
 5.60784340e+01 4.31372566e+01 6.35294113e+01 5.92156868e+01  
 3.76470604e+01 2.03921566e+01 3.13725491e+01 1.68627453e+01  
 2.62745094e+01 6.54901962e+01 5.49019623e+01 5.41176491e+01  
 4.70588226e+01 3.33333321e+01 1.76470585e+01 8.19607849e+01  
 8.62745094e+00 1.13725491e+01 8.23529434e+01 3.01960793e+01  
 6.23529434e+01 4.82352943e+01 4.62745094e+01 6.31372566e+01  
 2.43137264e+01 6.39215698e+01 4.23529434e+01 3.88235283e+01  
 5.25490189e+01 4.19607849e+01 2.74509811e+01 1.09803925e+01  
 9.41176510e+00 2.86274509e+01 5.05882378e+01 4.39215698e+01  
 4.11764717e+01 5.88235321e+01 4.58823547e+01 6.86274490e+01  
 3.80392151e+01 5.45098038e+01 4.43137245e+01 3.96078453e+01  
 7.52941208e+01 4.07843132e+01 6.74509811e+01 3.41176491e+01  
 8.11764755e+01 5.68627472e+01 6.03921585e+01 9.64705887e+01  
 7.49019623e+01 7.45098066e+00 2.27450981e+01 8.23529434e+00  
 9.80392170e+00 2.66666679e+01 3.68627472e+01 9.52941208e+01  
 5.80392151e+01 7.17647095e+01 6.15686302e+01 6.82352982e+01  
 7.68627472e+01 1.06666672e+02 5.09803925e+01 8.07843170e+01  
 6.50980377e+01 4.70588255e+00 7.76470566e+01 3.56862755e+01  
 7.05882339e+01 7.25490189e+01 7.84313726e+00 3.64705887e+01  
 5.72549019e+01 1.01960785e+02 6.19607849e+01 6.07843132e+01  
 5.17647057e+01 4.15686264e+01 7.72549057e+01 7.84313736e+01  
 6.62745132e+01 7.21568604e+01 1.18823532e+02 3.29411774e+01  
 6.70588226e+01 2.74509811e+00 6.43137283e+01 9.01960831e+01  
 6.66666641e+01 6.94117661e+01 6.58823547e+01 1.59607849e+02  
 5.49019623e+00 8.90196075e+01 8.58823547e+01 8.03921585e+01  
 7.37254944e+01 6.00000000e+01 7.56862793e+01 8.54901962e+01  
 9.49019623e+01 6.78431396e+01 6.98039246e+01 9.09803925e+01  
 1.78431381e+02 8.66666718e+01 7.09803925e+01 3.92156863e+00  
 6.90196075e+01 5.88235283e+00 9.72549057e+01 5.09803915e+00  
 7.41176453e+01 7.01960831e+01 7.64705887e+01 7.29411774e+01  
 3.52941179e+00 9.92156906e+01 1.24313728e+02 8.15686264e+01  
 7.45098038e+01 9.13725510e+01 7.33333359e+01 8.62745132e+01  
 7.60784302e+01 8.39215698e+01 7.80392151e+01 8.47058868e+01  
 8.86274490e+01 9.17647095e+01 6.66666698e+00 8.35294113e+01  
 1.03137253e+02 1.36862747e+02 8.78431396e+01 7.88235321e+01  
 1.03529411e+02 1.22352943e+02 7.96078415e+01 9.60784302e+01  
 1.25490196e+02 7.92156906e+01 1.01176476e+02 1.35294113e+02  
 7.05882359e+00 8.70588226e+01 1.13725494e+02 6.27450991e+00  
 8.94117661e+01 8.74509811e+01 8.27451019e+01 9.29411774e+01  
 9.88235321e+01 1.00784317e+02 9.45098038e+01 9.56862793e+01  
 8.50980377e+01 8.82352982e+01 1.00392159e+02 8.98039246e+01  
 9.37254944e+01 1.16078430e+02 9.25490189e+01 9.41176453e+01  
 1.12941177e+02 1.11372551e+02 1.31764709e+02 1.05098038e+02  
 8.31372528e+01 1.01568626e+02 1.12156868e+02 1.52549026e+02  
 9.68627472e+01 1.17647064e+02 1.04313728e+02 1.07058823e+02  
 1.32941177e+02 1.27843140e+02 1.65882355e+02 1.09019608e+02  
 9.84313736e+01 1.32549026e+02 9.05882339e+01 9.21568680e+01  
 1.14509804e+02 1.24705887e+02 1.02352943e+02 1.63529419e+02  
 1.10980392e+02 1.03921570e+02 9.33333359e+01 1.15686279e+02  
 1.34509811e+02 1.07450981e+02 1.09803925e+02 1.73435293e+04  
 1.32156860e+02 2.35294127e+00 1.68235291e+02 1.06274513e+02  
 1.38039215e+02 1.16862747e+02 1.08627449e+02 1.23921570e+02  
 9.80392151e+01 1.05490196e+02 1.08235298e+02 1.28235291e+02  
 1.09411766e+02 1.14901962e+02 1.69803925e+02 1.25882355e+02  
 9.96078415e+01 1.52941177e+02 1.07843140e+02 1.05882355e+02  
 1.14117645e+02 1.44705887e+02 1.41176468e+02 1.91764709e+02  
 1.30588242e+02 1.36078430e+02 1.15294121e+02 1.56862748e+00  
 1.00000000e+02 1.10196083e+02 1.41960785e+02 1.74901962e+02  
 1.26666672e+02 1.17254906e+02 1.19215691e+02 2.16470596e+02  
 9.76470642e+01 1.29019608e+02 1.12549019e+02 1.11764709e+02  
 1.49019608e+02 1.02745102e+02 1.61176468e+02 1.25098038e+02  
 1.45098038e+02 1.74117645e+02 1.58039215e+02 1.85882355e+02  
 1.38823532e+02 1.40784317e+02 2.30588242e+02 1.48627457e+02  
 1.96078432e+00 1.39215683e+02 1.36470596e+02 1.62352951e+02  
 1.21176476e+02 1.42352951e+02 1.60392166e+02 1.19607841e+02  
 1.04705887e+02 1.35686279e+02 1.56862747e+02 1.69019608e+02  
 1.49411774e+02 1.22745102e+02 1.70980392e+02 1.34901962e+02  
 1.40000000e+02 1.46274506e+02 1.10588234e+02 1.16470589e+02  
 1.13450986e+04 1.13333336e+02 1.20784317e+02 1.20392159e+02  
 1.21568626e+02 1.23529411e+02 1.33725494e+02 1.30196075e+02  
 1.43529419e+02 1.94117645e+02 1.31372559e+02 1.21960785e+02  
 1.54901962e+02 1.41568634e+02 1.29803925e+02 1.36141182e+04  
 1.23137260e+02 1.27058823e+02 1.20000000e+02 1.27450981e+02  
 1.55686279e+02 1.65098038e+02 1.30980392e+02 1.45972549e+04  
 1.45882355e+02 1.67450989e+02 1.90196075e+02 1.33333328e+02  
 1.42745102e+02 1.47058823e+02 1.68666660e+04 1.54117645e+02  
 1.48235291e+02 1.18039215e+02 1.84313736e+02 1.63137253e+02  
 1.43137253e+02 1.29411774e+02 1.26274513e+02 1.81568634e+02  
 1.47450989e+02 1.66274506e+02 1.70588242e+02 5.21568665e+02  
 1.18431374e+02 1.54509811e+02 1.52156860e+02 1.44313721e+02  
 1.37647064e+02 1.40392166e+02 1.90980392e+02 1.74509811e+02  
 2.41568634e+02 1.34117645e+02 1.73333344e+02 1.56470596e+02  
 1.78039215e+02 9.37843164e+03 1.72156860e+02 2.06666672e+02  
 1.75686279e+02 1.80784317e+02 1.89803925e+02 1.45490204e+02  
 1.55294113e+02 1.33874512e+04 1.50196075e+02 1.57647064e+02  
 1.47843140e+02 2.04313736e+02 1.43921570e+02 1.39607849e+02  
 1.95294128e+02 1.88627457e+02 1.59215683e+02 1.64705887e+02  
 1.49803925e+02 1.31431377e+04 1.86274506e+02 1.51372559e+02  
 2.06482363e+04 1.65639219e+04 1.58823532e+02 2.03529419e+02  
 1.24039219e+04 1.38431381e+02 1.66666672e+02 1.53333328e+02  
 1.88329414e+04 1.46666672e+02 1.28576475e+04 1.60345098e+04  
 1.93333344e+02 1.53725494e+02 5.43921570e+02 1.92403926e+04  
 2.13290195e+04 1.69411774e+02 1.77647064e+02 1.32360791e+04]  
Absolute Load[%]  
0.000000 310194  
14.901961 59745  
15.686275 58177  
14.117647 55841  
12.549020 55293  
 ...   
12403.921875 1  
13387.451172 1  
20648.236328 1  
177.647064 1  
13236.079102 1  
Name: count, Length: 463, dtype: int64  
  
Unique values in column 'OAT[DegC]':  
Number of unique values: 182  
[ nan 5. 6. 3. -1. 1. -14.75 0.75 13.5 -5.5  
 7. -4. 0. 14. -8.5 4. 6.5 -9. -11.5 -7.  
 6.25 2. -3. 7.5 -3.5 11. -2. -15.75 8. 15.  
 2.25 -6. -2.75 -16. -3.75 -5. -13. -11.25 4.25 -8.  
 -8.75 -2.5 5.5 -10. -1.5 12. 7.25 3.75 13. 2.5  
 2.75 -0.75 1.5 9. -6.5 1.75 6.75 0.25 -16.5 10.5  
 10. 20. -0.5 8.25 -13.25 -3.25 10.75 -12. 7.75 4.5  
 14.5 17. -10.75 0.5 9.75 16. 10.25 -1.75 8.5 -0.25  
 -10.5 -11. -7.5 9.25 -10.25 3.5 11.5 -9.25 -9.5 3.25  
 9.5 -6.75 -1.25 15.5 -6.25 5.75 -4.75 5.25 -14. -17.  
 -9.75 18. 23. 19. 28. -8.25 -7.25 -5.75 4.75 -12.5  
 -15. 11.75 -4.5 11.25 -7.75 -21. 8.75 -17.5 -13.75 -5.25  
 -12.25 -37. 12.75 24. -19.5 13.25 1.25 -14.5 -15.5 39.  
 -2.25 12.5 21. 13.75 -12.75 -11.75 15.25 -18. 16.5 -16.25  
 -4.25 27. -13.5 33. 14.75 -17.75 14.25 -39. 32. -14.25  
 35. 12.25 -20.25 -19. 18.5 26. -18.5 -20. 30. 16.75  
 34. 18.25 -24.5 22. 25. -40. 15.75 29. -20.75 -17.25  
 -15.25 -20.5 -16.75 17.5 17.75 16.25 -18.25 -21.25 -21.5 -38.  
 -36. -19.75 18.75]  
OAT[DegC]  
 0.00 116704  
 1.00 100805  
 4.00 97725  
 2.00 96557  
 3.00 93900  
 ...   
 17.50 25  
-36.00 21  
-18.25 16  
 18.75 7  
-19.75 1  
Name: count, Length: 182, dtype: int64  
  
Unique values in column 'Fuel Rate[L/hr]':  
Number of unique values: 2002  
[ nan 0. 5.58564663 ... 0.9301461 10.91371441  
 2.79520822]  
Fuel Rate[L/hr]  
0.000000 123691  
1.669493 393  
1.640873 363  
1.693343 274  
1.645643 262  
 ...   
2.480390 1  
0.930146 1  
9.640129 1  
2.876298 1  
2.795208 1  
Name: count, Length: 2002, dtype: int64  
  
Unique values in column 'Air Conditioning Power[kW]':  
Number of unique values: 50  
[ nan 0. 5.44000006 0.19999999 0.23999999 6.15999985  
 5.71999979 5.5999999 5.92000008 5.75999975 5.87999964 5.27999973  
 6.23999977 0.12 0.28 5.35999966 5.19999981 5.79999971  
 5.4000001 6. 6.19999981 5.63999987 5.55999994 0.16  
 0.35999998 6.11999989 6.07999992 5.51999998 0.47999999 5.67999983  
 0.31999999 0.44 5.23999977 0.39999998 5.83999968 5.96000004  
 4.19999981 0.04 5.31999969 6.27999973 6.03999996 5.15999985  
 5.48000002 4.83999968 0.51999998 0.56 0.59999996 4.75999975  
 6.31999969 0.08 6.35999966]  
Air Conditioning Power[kW]  
0.00 118246  
0.20 5768  
0.24 4017  
0.16 3285  
0.28 2527  
5.88 1969  
6.20 1925  
5.92 1721  
6.16 1629  
5.76 1347  
6.12 1259  
0.32 1239  
6.24 1235  
5.96 1162  
5.84 1110  
0.36 1045  
6.08 1003  
5.80 890  
5.36 741  
5.40 724  
5.72 687  
6.04 668  
0.40 648  
5.68 634  
6.00 557  
5.44 485  
0.12 428  
5.28 417  
5.52 335  
0.04 313  
5.56 288  
0.44 254  
5.64 225  
5.60 217  
5.32 157  
5.48 146  
5.20 129  
6.28 101  
0.52 96  
5.24 95  
0.48 88  
0.60 68  
4.84 55  
5.16 52  
4.20 47  
4.76 43  
6.32 41  
0.08 30  
6.36 17  
0.56 14  
Name: count, dtype: int64  
  
Unique values in column 'Air Conditioning Power[Watts]':  
Number of unique values: 58  
[ nan 0. 400. 350. 450. 200. 850. 1100. 950. 150. 550. 900.  
 1050. 1200. 600. 100. 300. 160. 1400. 120. 240. 250. 2050. 320.  
 700. 750. 1000. 1500. 2300. 40. 650. 1150. 800. 1800. 50. 500.  
 440. 80. 280. 1700. 1300. 480. 1950. 1850. 2500. 1450. 1250. 360.  
 1750. 1350. 1900. 2350. 680. 2250. 520. 640. 2000. 960. 560.]  
Air Conditioning Power[Watts]  
0.0 471679  
200.0 13533  
400.0 11769  
150.0 11439  
100.0 7712  
450.0 6928  
350.0 6901  
850.0 4224  
160.0 3737  
1050.0 3716  
900.0 3565  
950.0 3428  
1100.0 3335  
800.0 3141  
1000.0 2969  
240.0 2609  
120.0 2297  
750.0 2195  
280.0 2188  
650.0 1886  
250.0 1876  
700.0 1770  
300.0 1710  
320.0 1508  
600.0 1489  
500.0 1319  
1200.0 1133  
80.0 974  
550.0 842  
1150.0 762  
50.0 531  
1300.0 471  
1850.0 460  
1800.0 395  
1250.0 315  
440.0 291  
2050.0 283  
360.0 268  
1700.0 238  
1900.0 185  
1350.0 166  
1950.0 142  
2500.0 118  
480.0 116  
1750.0 110  
1400.0 94  
40.0 88  
2300.0 61  
2250.0 61  
1450.0 61  
1500.0 56  
640.0 46  
960.0 40  
2000.0 32  
2350.0 28  
520.0 17  
680.0 6  
560.0 6  
Name: count, dtype: int64  
  
Unique values in column 'Heater Power[Watts]':  
Number of unique values: 18  
[ nan 0. 3500. 1250. 750. 250. 500. 3000. 1500. 1000. 2000. 3750.  
 2500. 3250. 2250. 2750. 1750. 4000. 4250.]  
Heater Power[Watts]  
0.0 32523  
250.0 11992  
500.0 9871  
750.0 6786  
1000.0 3882  
1250.0 2630  
1500.0 2331  
3750.0 2298  
3000.0 2250  
1750.0 2241  
3500.0 2211  
3250.0 1761  
2750.0 1629  
2250.0 1539  
2500.0 1533  
2000.0 1402  
4000.0 717  
4250.0 9  
Name: count, dtype: int64  
  
Unique values in column 'HV Battery Current[A]':  
Number of unique values: 16107  
[ nan -1.5 -8.25 ... 44.82998657 -122.60009766  
 -120.72000122]  
HV Battery Current[A]  
-1.500000 3996  
-3.000000 2284  
-4.500000 2133  
 0.000000 2121  
-6.000000 1974  
 ...   
-89.300049 1  
-158.600037 1  
-153.600006 1  
 112.839996 1  
-112.300049 1  
Name: count, Length: 16107, dtype: int64  
  
Unique values in column 'HV Battery SOC[%]':  
Number of unique values: 5827  
[ nan 0. 58.03923416 ... 74.36708832 96.2784729  
 47.34177017]  
HV Battery SOC[%]  
0.000000 216097  
2.000000 27003  
1.000000 26408  
3.000000 17299  
0.392157 10698  
 ...   
59.177216 1  
43.746830 1  
53.822784 1  
38.544304 1  
37.126579 1  
Name: count, Length: 5827, dtype: int64  
  
Unique values in column 'HV Battery Voltage[V]':  
Number of unique values: 1019  
[ nan 198. 361.625 ... 318.125 262.5 309.75 ]  
HV Battery Voltage[V]  
197.000 26701  
197.500 17229  
198.000 14372  
195.500 12667  
198.500 10976  
 ...   
395.875 2  
316.875 1  
256.000 1  
318.125 1  
309.750 1  
Name: count, Length: 1019, dtype: int64  
  
Unique values in column 'Short Term Fuel Trim Bank 1[%]':  
Number of unique values: 121  
[ -0.78125 -1.5625 0. -7.03125 nan -2.34375 -3.90625  
 7.8125 2.34375 -5.46875 11.71875 5.46875 -18.75 1.5625  
 0.78125 3.90625 13.28125 -9.375 -6.25 3.125 -11.71875  
 18.75 7.03125 -10.15625 4.6875 12.5 -8.59375 8.59375  
 10.15625 -3.125 -4.6875 21.09375 9.375 6.25 -7.8125  
 -17.96875 15.625 14.0625 -10.9375 10.9375 -15.625 19.53125  
 -13.28125 -12.5 28.90625 16.40625 30.46875 32.03125 23.4375  
 14.84375 -14.84375 46.875 -14.0625 -21.09375 -16.40625 -20.3125  
 25. -17.1875 25.78125 21.875 -28.90625 17.1875 17.96875  
 -25. 26.5625 34.375 -26.5625 22.65625 24.21875 50.78125  
 29.6875 -19.53125 28.125 -33.59375 35.15625 41.40625 43.75  
 -27.34375 31.25 -29.6875 20.3125 32.8125 51.5625 27.34375  
 -21.875 50. -22.65625 42.1875 36.71875 49.21875 33.59375  
 35.9375 53.125 44.53125 40.625 37.5 42.96875 38.28125  
 -24.21875 39.84375 -23.4375 39.0625 52.34375 46.09375 -28.125  
 -25.78125 45.3125 47.65625 57.8125 76.5625 -32.03125 78.90625  
 88.28125 -31.25 72.65625 81.25 89.84375 -34.375 -35.15625  
 73.4375 48.4375 53.90625]  
Short Term Fuel Trim Bank 1[%]  
 0.00000 1044225  
 0.78125 301327  
-0.78125 293847  
-1.56250 269404  
 1.56250 254044  
 ...   
 76.56250 3  
-33.59375 2  
-34.37500 1  
 48.43750 1  
 53.90625 1  
Name: count, Length: 121, dtype: int64  
  
Unique values in column 'Short Term Fuel Trim Bank 2[%]':  
Number of unique values: 108  
[ nan 0.78125 0. -3.125 -2.34375 1.5625 3.125  
 -0.78125 -1.5625 -8.59375 -6.25 7.03125 3.90625 5.46875  
 -3.90625 4.6875 -7.8125 2.34375 -7.03125 8.59375 -4.6875  
 9.375 -5.46875 -9.375 6.25 11.71875 10.15625 18.75  
 -10.15625 -12.5 -93.75 13.28125 -10.9375 28.90625 7.8125  
 -18.75 14.84375 15.625 10.9375 12.5 19.53125 24.21875  
 -16.40625 -28.90625 17.96875 22.65625 31.25 -11.71875 21.875  
 26.5625 -14.0625 -29.6875 25. 50.78125 30.46875 25.78125  
 32.03125 -17.96875 16.40625 45.3125 43.75 20.3125 17.1875  
 -17.1875 -21.09375 -15.625 33.59375 39.0625 32.8125 37.5  
 -13.28125 -14.84375 14.0625 -22.65625 29.6875 51.5625 -20.3125  
 27.34375 -19.53125 23.4375 28.125 34.375 21.09375 36.71875  
 53.125 -27.34375 53.90625 41.40625 35.15625 42.96875 35.9375  
 -25. 42.1875 -23.4375 -21.875 44.53125 52.34375 40.625  
 -24.21875 50. 39.84375 49.21875 -26.5625 -25.78125 -30.46875  
 38.28125 46.875 47.65625 46.09375]  
Short Term Fuel Trim Bank 2[%]  
 0.00000 192928  
 0.78125 90488  
-0.78125 90185  
-1.56250 85074  
 1.56250 80793  
 ...   
-30.46875 6  
 47.65625 6  
-26.56250 5  
 46.87500 5  
 46.09375 2  
Name: count, Length: 108, dtype: int64  
  
Unique values in column 'Long Term Fuel Trim Bank 1[%]':  
Number of unique values: 96  
[ 2.34375 0. -1.5625 -0.78125 4.6875 nan 6.25  
 5.46875 3.125 3.90625 7.8125 12.5 1.5625 -3.125  
 0.78125 -4.6875 7.03125 -5.46875 -7.8125 9.375 -2.34375  
 14.0625 -11.71875 18.75 -7.03125 -3.90625 -13.28125 8.59375  
 17.1875 -6.25 17.96875 10.15625 -10.15625 21.09375 -23.4375  
 10.9375 -12.5 -9.375 -16.40625 11.71875 25. 16.40625  
 -8.59375 -15.625 -10.9375 15.625 14.84375 23.4375 -17.96875  
 24.21875 13.28125 19.53125 -14.0625 -14.84375 -17.1875 22.65625  
 21.875 38.28125 -20.3125 -18.75 20.3125 37.5 29.6875  
 25.78125 30.46875 -27.34375 -19.53125 28.125 39.84375 -21.875  
 39.0625 45.3125 36.71875 -24.21875 35.9375 43.75 -28.125  
 -22.65625 26.5625 28.90625 31.25 -21.09375 40.625 32.03125  
 27.34375 -25.78125 33.59375 -30.46875 44.53125 -25. 35.15625  
 -32.8125 34.375 32.8125 -26.5625 -29.6875 42.96875]  
Long Term Fuel Trim Bank 1[%]  
 0.00000 332021  
-0.78125 295995  
-1.56250 294897  
 0.78125 279480  
 1.56250 278046  
 ...   
-25.00000 14  
 40.62500 8  
 44.53125 7  
-29.68750 5  
 42.96875 4  
Name: count, Length: 96, dtype: int64  
  
Unique values in column 'Long Term Fuel Trim Bank 2[%]':  
Number of unique values: 68  
[ nan -1.5625 2.34375 0.78125 0. 7.8125 15.625  
 1.5625 -0.78125 5.46875 6.25 -4.6875 -10.15625 3.125  
 7.03125 3.90625 -2.34375 4.6875 10.9375 -7.03125 -3.90625  
 -5.46875 -3.125 10.15625 14.84375 -6.25 21.875 -14.84375  
 11.71875 16.40625 -15.625 -9.375 13.28125 9.375 -8.59375  
 -12.5 25. -10.9375 8.59375 12.5 -7.8125 14.0625  
 -16.40625 23.4375 -13.28125 17.1875 19.53125 -11.71875 -14.0625  
 17.96875 20.3125 -20.3125 21.09375 18.75 -17.1875 -22.65625  
 22.65625 -18.75 24.21875 -25. -17.96875 -27.34375 -21.09375  
 -19.53125 -30.46875 -21.875 -24.21875 -25.78125 -26.5625 ]  
Long Term Fuel Trim Bank 2[%]  
 0.00000 100367  
-0.78125 77210  
 0.78125 74998  
 1.56250 71533  
 2.34375 67680  
 ...   
-25.00000 42  
-27.34375 31  
-21.87500 20  
-24.21875 14  
-26.56250 2  
Name: count, Length: 68, dtype: int64  
  
Unique values in column 'Vehicle Type':  
Number of unique values: 4  
['ICE' 'HEV' 'PHEV' 'EV']  
Vehicle Type  
ICE 3185437  
HEV 1185042  
PHEV 660394  
EV 87605  
Name: count, dtype: int64  
  
Unique values in column 'Vehicle Class':  
Number of unique values: 2  
[nan 'Car' 'SUV']  
Vehicle Class  
Car 789552  
SUV 731  
Name: count, dtype: int64  
  
Unique values in column 'Engine Configuration & Displacement':  
Number of unique values: 70  
['4-FI 2.4L' '6-GAS/ELECTRIC 3.5L' '4-GAS/ELECTRIC 1.5L'  
 '4-GAS/ELECTRIC 1.8L' '6-FI 3.5L' '4-GAS/ELECTRIC 1.4L'  
 '4-FI 1.3L GAS/ELEC.' '4-GAS/ELECTRIC 2.3L' 'ELECTRIC' '8-FI 5.3L'  
 '4-GAS/ELECTRIC 2.0L' '4-FI 1.8L' '6-FI 3.0L' '4-FI 2.3L ULEV'  
 '4-FI 2.0L' '6-FI 3.3L' '4-FI 1.5L' '6-FI 3.6L' '4-FI 2.5L'  
 '4-FI 2.3L T/C' '4-FI 1.6L T/C' '4-FI 2.0L T/C' '8-4V/FI 6.0L'  
 '4-FI 1.6L' '4-FI S/C 1.8L GAS' '8-FI 5.7L HEMI (Hemi engine)'  
 '4-FI 2.0L PZEV' '4-GAS/ELECTRIC 2.5L' '6-FI 3.7L' '4-GAS/ELECTRIC 2.4L'  
 '4-FI T/C 1.4L' '4-FI 1.4L T/C' '6-EFI 4.2L ' 'V6 3.5L' '4-FI 2.2L'  
 '2.4L' '6-FI 4.3L' '6-EFI 3.0L' '5-FI 2.5L' '6-GAS/ELECTRIC 3.3L'  
 '8-FI 4.6L' 'V6 3.1L' '3.0L 6cyl 4A' 'H-4 2.0 L/122' '6-242-MFI 4.0L'  
 '6-FI 2.7L' '8-FI 4.8L' '10-FI 6.8L' '8-FI 5.7L HEMI' 'V6 3.8L'  
 '8-FI 5.4L' 'V8 4.7L' 'V6 4.0L' '4-FI T/C 2.0L' 'V6 3.0L' '6-FI 3.4L'  
 'I4 2.2L' '5-FI 2.5L PZEV' '4-FI 1.5L T/C' '6-FI 4.2L' '6-FI 3.1L'  
 '4-FI 2.3L' 'I4 2.4L Flex Fuel' '4-GAS/ELECTRIC 1.6L' '6-FI 3.8L'  
 '2.3L Gasoline I4' '8-FI 4.7L' '3-FI 1.0L GAS/ELEC.' '8-EFI 5.0L'  
 '8-DSL 6.7L T/C']  
Engine Configuration & Displacement  
4-GAS/ELECTRIC 1.8L 648290  
6-FI 3.5L 473533  
4-FI 2.4L 471870  
4-FI 2.5L 428581  
4-GAS/ELECTRIC 1.5L 419021  
 ...   
8-FI 4.7L 2152  
6-FI 3.1L 1978  
8-EFI 5.0L 714  
4-GAS/ELECTRIC 1.6L 581  
8-DSL 6.7L T/C 121  
Name: count, Length: 70, dtype: int64  
  
Unique values in column 'Transmission':  
Number of unique values: 16  
[nan 'CVT' '6-SP ECT AUTOMATIC' '5-SP AUTOMATIC' '4-SP AUTOMATIC'  
 'AUTOMATIC/CVT' '5-SP MANUAL' 'FULL TIME 4WD MANUAL' '4-SP Automatic'  
 'AUTOMATIC' '5-SP ECT AUTOMATIC' '6-SP AUTOMATIC' '5-SP Automatic'  
 '5-SP AWD MANUAL' 'FULL TIME 4WD AUTOMATIC' '9-SP Automatic'  
 '6-SP AWD MANUAL']  
Transmission  
CVT 697687  
5-SP AUTOMATIC 299707  
AUTOMATIC/CVT 227330  
4-SP AUTOMATIC 70071  
4-SP Automatic 52391  
AUTOMATIC 43605  
5-SP MANUAL 41649  
6-SP AUTOMATIC 36530  
6-SP ECT AUTOMATIC 27782  
5-SP ECT AUTOMATIC 21522  
FULL TIME 4WD MANUAL 15006  
FULL TIME 4WD AUTOMATIC 13287  
9-SP Automatic 8844  
5-SP AWD MANUAL 8271  
5-SP Automatic 8147  
6-SP AWD MANUAL 3288  
Name: count, dtype: int64  
  
Unique values in column 'Drive Wheels':  
Number of unique values: 1  
[nan 'FWD']  
Drive Wheels  
FWD 747999  
Name: count, dtype: int64  
  
Unique values in column 'Generalized\_Weight':  
Number of unique values: 10  
[3000. 4500. 3500. 4000. 2500. 5000. 6000. 5500. nan 2000. 6500.]  
Generalized\_Weight  
3500.0 1499991  
3000.0 1484692  
4000.0 875236  
2500.0 540634  
4500.0 481850  
5000.0 51999  
5500.0 36883  
2000.0 3695  
6000.0 3122  
6500.0 121  
Name: count, dtype: int64

df.head().to\_dict()

{'DayNum': {0: 130.55213663,  
 1: 109.902064944,  
 2: 46.5006110926,  
 3: 46.7281407963,  
 4: 94.7527253241},  
 'VehId': {0: 487, 1: 392, 2: 285, 3: 566, 4: 242},  
 'Trip': {0: 1168, 1: 637, 2: 1387, 3: 216, 4: 600},  
 'Timestamp(ms)': {0: 35200, 1: 343300, 2: 1287100, 3: 76300, 4: 70900},  
 'Latitude[deg]': {0: 42.25612,  
 1: 42.2944894444,  
 2: 42.2817447222,  
 3: 42.3052466667,  
 4: 42.2303102778},  
 'Longitude[deg]': {0: -83.6924769444,  
 1: -83.6826069444,  
 2: -83.7568280556,  
 3: -83.735115,  
 4: -83.6957636111},  
 'Vehicle Speed[km/h]': {0: 58.0, 1: 63.0, 2: 49.0, 3: 51.0, 4: 63.0},  
 'MAF[g/sec]': {0: 5.53999996185,  
 1: 13.3999996185,  
 2: 17.5200004578,  
 3: 0.070000000298,  
 4: 0.930000007153},  
 'Engine RPM[RPM]': {0: 1504.0, 1: 1135.0, 2: 1665.0, 3: 0.0, 4: 0.0},  
 'Absolute Load[%]': {0: nan,  
 1: 35.2941169739,  
 2: 44.7058830261,  
 3: 0.0,  
 4: 56.4705886841},  
 'OAT[DegC]': {0: nan, 1: 5.0, 2: nan, 3: nan, 4: 5.0},  
 'Fuel Rate[L/hr]': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'Air Conditioning Power[kW]': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'Air Conditioning Power[Watts]': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'Heater Power[Watts]': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'HV Battery Current[A]': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'HV Battery SOC[%]': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'HV Battery Voltage[V]': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'Short Term Fuel Trim Bank 1[%]': {0: -0.78125,  
 1: -0.78125,  
 2: -1.5625,  
 3: 0.0,  
 4: -7.03125},  
 'Short Term Fuel Trim Bank 2[%]': {0: nan,  
 1: 0.78125,  
 2: nan,  
 3: nan,  
 4: nan},  
 'Long Term Fuel Trim Bank 1[%]': {0: 2.34375,  
 1: 0.0,  
 2: -1.5625,  
 3: -0.78125,  
 4: 4.6875},  
 'Long Term Fuel Trim Bank 2[%]': {0: nan, 1: -1.5625, 2: nan, 3: nan, 4: nan},  
 'Vehicle Type': {0: 'ICE', 1: 'HEV', 2: 'ICE', 3: 'HEV', 4: 'HEV'},  
 'Vehicle Class': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'Engine Configuration & Displacement': {0: '4-FI 2.4L',  
 1: '6-GAS/ELECTRIC 3.5L',  
 2: '4-FI 2.4L',  
 3: '4-GAS/ELECTRIC 1.5L',  
 4: '4-GAS/ELECTRIC 1.8L'},  
 'Transmission': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'Drive Wheels': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'Generalized\_Weight': {0: 3000.0, 1: 4500.0, 2: 3000.0, 3: 3000.0, 4: 3500.0}}

'''  
Mapped these values to OAT[DegC] values and create a new column 'OAT\_Category' (< -20 'Extremely Cold' , -20 to 0 'Cold', 0 to 10 'Cool', 10 to 20 'Mild', 20 to 30 'Warm', > 30 'Hot')  
'''  
  
def categorize\_oat(value):  
 if value < -20:  
 return 'Extremely Cold'  
 elif -20 <= value < 0:  
 return 'Cold'  
 elif 0 <= value < 10:  
 return 'Cool'  
 elif 10 <= value < 20:  
 return 'Mild'  
 elif 20 <= value < 30:  
 return 'Warm'  
 elif value >= 30:  
 return 'Hot'  
 else:  
 return np.nan  
  
df['OAT\_Category'] = df['OAT[DegC]'].apply(categorize\_oat)

df['OAT\_Category'].unique()

array([nan, 'Cool', 'Cold', 'Mild', 'Warm', 'Extremely Cold', 'Hot'],  
 dtype=object)

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5118478 entries, 0 to 5118477  
Data columns (total 29 columns):  
 # Column Dtype   
--- ------ -----   
 0 DayNum float64  
 1 VehId int64   
 2 Trip int64   
 3 Timestamp(ms) int64   
 4 Latitude[deg] float64  
 5 Longitude[deg] float64  
 6 Vehicle Speed[km/h] float64  
 7 MAF[g/sec] float64  
 8 Engine RPM[RPM] float64  
 9 Absolute Load[%] float64  
 10 OAT[DegC] float64  
 11 Fuel Rate[L/hr] float64  
 12 Air Conditioning Power[kW] float64  
 13 Air Conditioning Power[Watts] float64  
 14 Heater Power[Watts] float64  
 15 HV Battery Current[A] float64  
 16 HV Battery SOC[%] float64  
 17 HV Battery Voltage[V] float64  
 18 Short Term Fuel Trim Bank 1[%] float64  
 19 Short Term Fuel Trim Bank 2[%] float64  
 20 Long Term Fuel Trim Bank 1[%] float64  
 21 Long Term Fuel Trim Bank 2[%] float64  
 22 Vehicle Type object   
 23 Vehicle Class object   
 24 Engine Configuration & Displacement object   
 25 Transmission object   
 26 Drive Wheels object   
 27 Generalized\_Weight float64  
 28 OAT\_Category object   
dtypes: float64(20), int64(3), object(6)  
memory usage: 1.1+ GB

'''  
Converted DayNum to timestamp using vectorized timedelta to create separate columns for df['DateTime'], df['Date'] and df['Time']  
'''  
  
import pandas as pd  
from datetime import datetime, timedelta  
  
# Reference datetime for DayNum = 1  
reference\_date = datetime(2017, 11, 1)  
  
# Convert DayNum to timestamp using vectorized timedelta  
df['DateTime'] = pd.to\_timedelta(df['DayNum'] - 1, unit='D') + reference\_date  
  
# Create separate columns for date and time  
df['Date'] = df['DateTime'].dt.date  
  
# Create a time column with proper formatting  
df['Time'] = df['DateTime'].dt.time

'''  
Calculated distance travelled in km using Vehicle Speed[km/h] and Timestamp(ms) - df['Distance[km]'] : df['Distance[km]'] = df['Vehicle Speed[km/h]'] \* (df['Timestamp(ms)'] / 3600000)  
'''  
  
df['Distance[km]'] = df['Vehicle Speed[km/h]'] \* (df['Timestamp(ms)'] / 3600000)

'''  
Calculated FCR based on the Algorithm in IEEE paper - [the VED paper](https://arxiv.org/abs/1905.02081)  
  
Algorithm 1: Estimation of Fuel Consumption Rate (FCR)  
Input : FuelRate, MAF, AbsLoad, Displacementeng,  
RPMeng, ST FT, LT FT, AFR, ρair  
Output: FCR  
1 correction = (1 + ST FT/100 + LT FT/100)/AFR  
2 if FuelRate is available then  
3 return FuelRate  
4 else if MAF is available then  
5 return MAF \* correction  
6 else if AbsLoad and RPMeng are available then  
7 MAF =  
AbsLoad/100\*ρair\*Displacementeng\*RPMeng/120  
8 return MAF \* correction  
9 else  
10 return NaN  
'''  
  
import pandas as pd  
import numpy as np  
  
# Constants  
AFR = 14.7 # typical AFR for gasoline engines  
ρ\_air = 1.184 # air density in kg/m³  
  
def compute\_fcr(df):  
 # Parse displacement in liters from 'Engine Configuration & Displacement' if format like "I4 2.0L"  
 def extract\_displacement(val):  
 try:  
 return float(val.split()[-1].replace("L", ""))  
 except:  
 return np.nan  
  
 df['Displacement\_L'] = df['Engine Configuration & Displacement'].apply(extract\_displacement)  
  
 # Compute correction factor  
 df['correction'] = (1 + df['Short Term Fuel Trim Bank 1[%]']/100 + df['Long Term Fuel Trim Bank 1[%]']/100) / AFR  
  
 # Step 1: Use FuelRate if available  
 df['FCR'] = np.where(  
 ~df['Fuel Rate[L/hr]'].isna(),  
 df['Fuel Rate[L/hr]'],  
 np.nan  
 )  
  
 # Step 2: Else if MAF is available  
 maf\_condition = df['FCR'].isna() & ~df['MAF[g/sec]'].isna()  
 df.loc[maf\_condition, 'FCR'] = df.loc[maf\_condition, 'MAF[g/sec]'] \* df.loc[maf\_condition, 'correction']  
  
 # Step 3: Else if AbsLoad and RPMeng are available  
 derived\_condition = df['FCR'].isna() & ~df['Absolute Load[%]'].isna() & ~df['Engine RPM[RPM]'].isna() & ~df['Displacement\_L'].isna()  
 maf\_derived = (df['Absolute Load[%]'] / 100) \* ρ\_air \* df['Displacement\_L'] \* df['Engine RPM[RPM]'] / 120  
 df.loc[derived\_condition, 'FCR'] = maf\_derived[derived\_condition] \* df.loc[derived\_condition, 'correction']  
  
 return df  
  
df = compute\_fcr(df)

df.columns  
# Save the final DataFrame to a CSV file

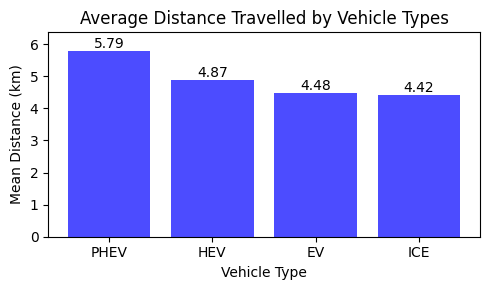
Index(['DayNum', 'VehId', 'Trip', 'Timestamp(ms)', 'Latitude[deg]',  
 'Longitude[deg]', 'Vehicle Speed[km/h]', 'MAF[g/sec]',  
 'Engine RPM[RPM]', 'Absolute Load[%]', 'OAT[DegC]', 'Fuel Rate[L/hr]',  
 'Air Conditioning Power[kW]', 'Air Conditioning Power[Watts]',  
 'Heater Power[Watts]', 'HV Battery Current[A]', 'HV Battery SOC[%]',  
 'HV Battery Voltage[V]', 'Short Term Fuel Trim Bank 1[%]',  
 'Short Term Fuel Trim Bank 2[%]', 'Long Term Fuel Trim Bank 1[%]',  
 'Long Term Fuel Trim Bank 2[%]', 'Vehicle Type', 'Vehicle Class',  
 'Engine Configuration & Displacement', 'Transmission', 'Drive Wheels',  
 'Generalized\_Weight', 'OAT\_Category', 'DateTime', 'Date', 'Time',  
 'Distance[km]', 'Displacement\_L', 'correction', 'FCR'],  
 dtype='object')

# 4 Sample plots

'''  
Average Distance Travelled by Vehicle Types  
'''  
  
df\_distance = df.groupby(['Vehicle Type'])['Distance[km]'].mean().reset\_index().sort\_values(by='Distance[km]', ascending=False)  
df\_distance

Vehicle Type Distance[km]  
3 PHEV 5.789103  
1 HEV 4.873559  
0 EV 4.478784  
2 ICE 4.421037

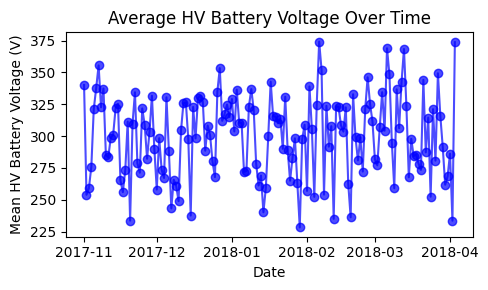
import matplotlib.pyplot as plt  
plt.figure(figsize=(5, 3))  
plt.bar(df\_distance['Vehicle Type'], df\_distance['Distance[km]'], alpha=0.7, color= 'blue')  
plt.title('Average Distance Travelled by Vehicle Types')  
plt.xlabel('Vehicle Type')  
plt.ylabel('Mean Distance (km)')  
plt.ylim(0, df\_distance['Distance[km]'].max() \* 1.1) # Set y-axis limit to 10% above max distance  
# Add text labels on top of the bars  
for index, value in enumerate(df\_distance['Distance[km]']):  
 plt.text(index, value , f"{value:.2f}", ha='center', va='bottom')  
plt.tight\_layout()  
plt.show()



'''  
Average HV Battery Voltage Over Time  
'''  
  
# Ensure 'Date' column is datetime type  
df['Date'] = pd.to\_datetime(df['Date'])  
  
# Group by day and calculate mean HV Battery Voltage  
df\_eot = df.groupby(df['Date'].dt.to\_period('D'))['HV Battery Voltage[V]'].mean().reset\_index()  
df\_eot['Date'] = df\_eot['Date'].dt.to\_timestamp()  
df\_eot = df\_eot.sort\_values(by='Date')  
df\_eot

Date HV Battery Voltage[V]  
0 2017-11-01 340.125991  
1 2017-11-02 253.911069  
2 2017-11-03 259.114447  
3 2017-11-04 276.070132  
4 2017-11-05 321.144065  
.. ... ...  
149 2018-03-30 261.779422  
150 2018-03-31 268.318241  
151 2018-04-01 286.188751  
152 2018-04-02 233.540929  
153 2018-04-03 373.718609  
  
[154 rows x 2 columns]

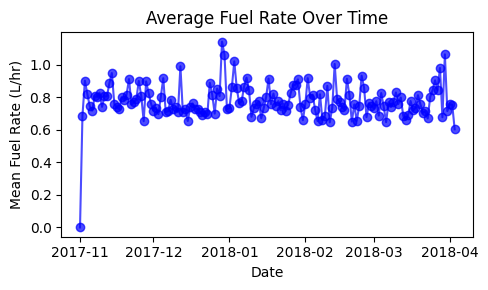
import matplotlib.pyplot as plt  
  
plt.figure(figsize=(5, 3))  
plt.plot(df\_eot['Date'], df\_eot['HV Battery Voltage[V]'], marker='o', linestyle='-', color='blue', alpha=0.7)  
plt.title('Average HV Battery Voltage Over Time')  
plt.xlabel('Date')  
plt.ylabel('Mean HV Battery Voltage (V)')  
plt.tight\_layout()  
plt.show()



'''  
Average Fuel Rate Over Time  
'''  
  
# Ensure 'Date' column is datetime type  
df['Date'] = pd.to\_datetime(df['Date'])  
  
# Group by day and calculate mean Fuel Rate  
df\_eof = df.groupby(df['Date'].dt.to\_period('D'))['FCR'].mean().reset\_index()  
df\_eof['Date'] = df\_eof['Date'].dt.to\_timestamp()  
df\_eof = df\_eof.sort\_values(by='Date')  
df\_eof

Date FCR  
0 2017-11-01 0.000000  
1 2017-11-02 0.687202  
2 2017-11-03 0.902463  
3 2017-11-04 0.819851  
4 2017-11-05 0.744445  
.. ... ...  
149 2018-03-30 1.064590  
150 2018-03-31 0.715263  
151 2018-04-01 0.757474  
152 2018-04-02 0.755047  
153 2018-04-03 0.603492  
  
[154 rows x 2 columns]

import matplotlib.pyplot as plt  
  
plt.figure(figsize=(5, 3))  
plt.plot(df\_eof['Date'], df\_eof['FCR'], marker='o', linestyle='-', color='blue', alpha=0.7)  
plt.title('Average Fuel Rate Over Time')  
plt.xlabel('Date')  
plt.ylabel('Mean Fuel Rate (L/hr)')  
plt.tight\_layout()  
plt.show()



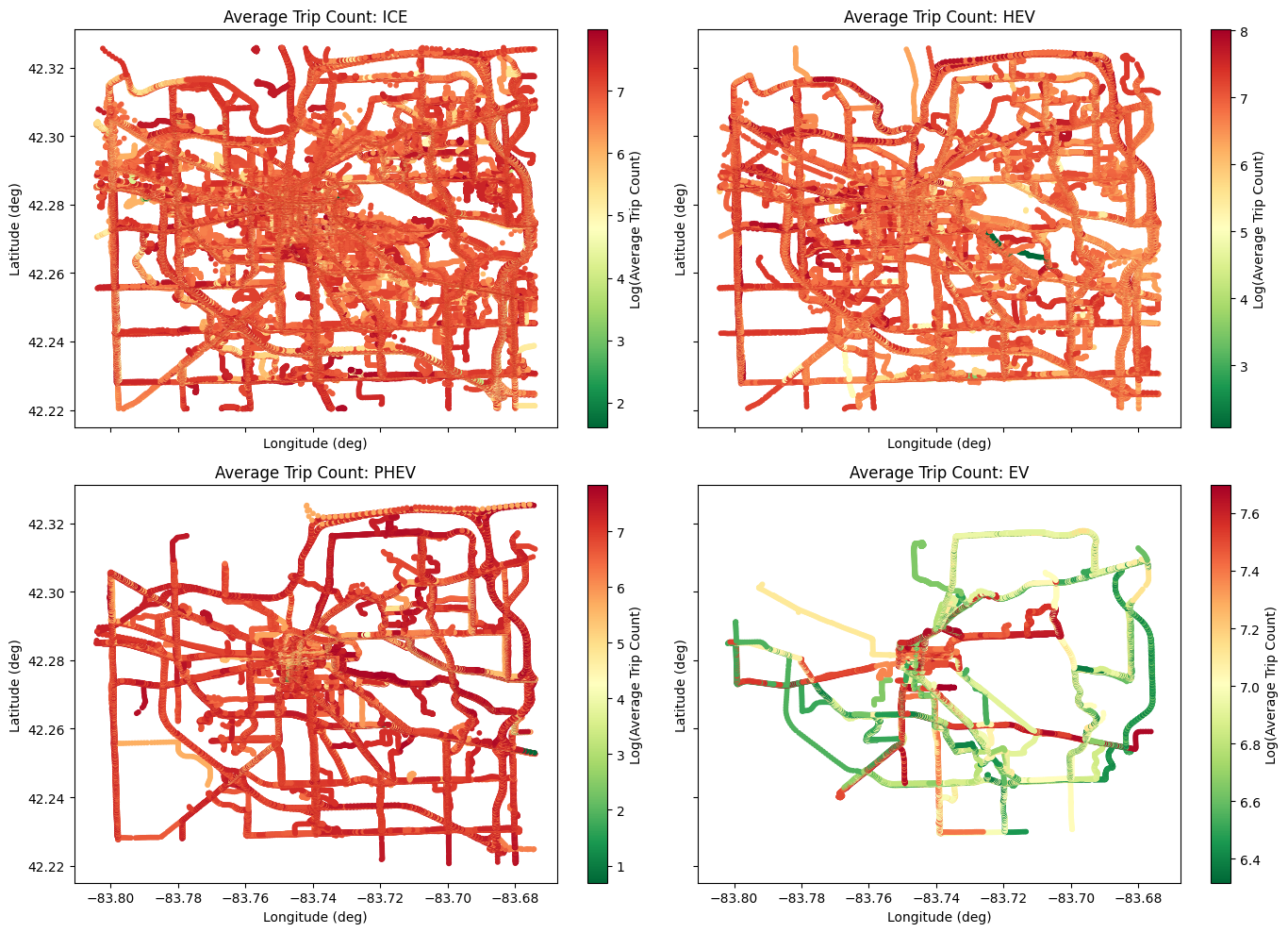
'''  
Average Trip Count by Location (Heat Map)  
'''  
  
df\_map = df.groupby(['Latitude[deg]', 'Longitude[deg]','Vehicle Type'])['Trip'].mean().reset\_index()  
df\_map

Latitude[deg] Longitude[deg] Vehicle Type Trip  
0 42.220305 -83.760323 ICE 1661.0  
1 42.220306 -83.767207 HEV 1502.0  
2 42.220309 -83.778162 ICE 1659.0  
3 42.220316 -83.734527 ICE 1955.0  
4 42.220316 -83.760919 ICE 1399.0  
... ... ... ... ...  
1856224 42.325775 -83.749506 ICE 1983.0  
1856225 42.325780 -83.756816 ICE 1983.0  
1856226 42.325787 -83.707453 ICE 1183.0  
1856227 42.325796 -83.749125 ICE 1089.0  
1856228 42.325800 -83.683371 ICE 1345.0  
  
[1856229 rows x 4 columns]

df\_map.head()

Latitude[deg] Longitude[deg] Vehicle Type Trip  
0 42.220305 -83.760323 ICE 1661.0  
1 42.220306 -83.767207 HEV 1502.0  
2 42.220309 -83.778162 ICE 1659.0  
3 42.220316 -83.734527 ICE 1955.0  
4 42.220316 -83.760919 ICE 1399.0

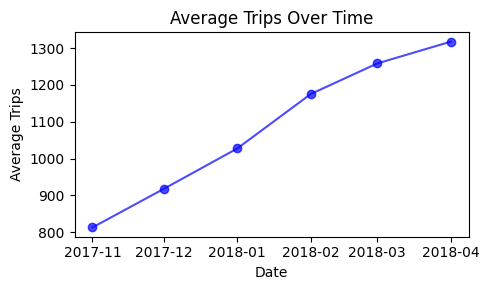
'''  
The code visualized the average trip count by location for each vehicle type using a scatter plot on subplots.   
It iterated over unique vehicle types, filtered the data for each type, and plotted longitude and latitude with color representing the logarithm of the average trip count.   
It set subplot titles and axis labels, and added a colorbar for each subplot.   
Finally, it adjusted the layout and displayed the figure.  
'''  
  
import matplotlib.pyplot as plt  
  
vehicle\_types = df\_map['Vehicle Type'].unique()  
fig, axes = plt.subplots(2, 2, figsize=(14, 10), sharex=True, sharey=True)  
axes = axes.flatten()  
  
for idx, vtype in enumerate(vehicle\_types):  
 ax = axes[idx]  
 data = df\_map[df\_map['Vehicle Type'] == vtype]  
 sc = ax.scatter(  
 data['Longitude[deg]'],  
 data['Latitude[deg]'],  
 c=np.log(data['Trip']),  
 marker='o',  
 s=10,  
 cmap='RdYlGn\_r'  
 )  
 ax.set\_title(f'Average Trip Count: {vtype}')  
 ax.set\_xlabel('Longitude (deg)')  
 ax.set\_ylabel('Latitude (deg)')  
 plt.colorbar(sc, ax=ax, label='Log(Average Trip Count)')  
  
plt.tight\_layout()  
plt.show()



'''  
Average Trips Over Time  
'''  
  
# Ensure 'Date' column is datetime type  
df['Date'] = pd.to\_datetime(df['Date'])  
  
# Group by month and calculate average trips per month  
df\_trip = df.groupby(df['Date'].dt.to\_period('M'))['Trip'].mean().reset\_index()  
df\_trip['Date'] = df\_trip['Date'].dt.to\_timestamp()  
df\_trip = df\_trip.sort\_values(by='Date')  
df\_trip

Date Trip  
0 2017-11-01 812.870813  
1 2017-12-01 917.432321  
2 2018-01-01 1027.306361  
3 2018-02-01 1175.387338  
4 2018-03-01 1258.388631  
5 2018-04-01 1317.635415

import matplotlib.pyplot as plt  
  
plt.figure(figsize=(5, 3))  
plt.plot(df\_trip['Date'], df\_trip['Trip'], marker='o', linestyle='-', color='blue', alpha=0.7)  
plt.title('Average Trips Over Time')  
plt.xlabel('Date')  
plt.ylabel('Average Trips')  
plt.tight\_layout()  
plt.show()



# 5 Exploratory Data Analysis

'''  
Calculated Power using Voltage and Current: df['HV Battery Power[Watts]'] = df['HV Battery Voltage[V]'] \* df['HV Battery Current[A]']  
'''  
  
df['HV Battery Power[Watts]'] = df['HV Battery Voltage[V]'] \* df['HV Battery Current[A]']

'''  
Data exploration of Battery Power, Fuel Consumption Rate (FCR) and Battery SOC using the features Battery Power, AC Power, Heater Power with respect to OAT\_Category and Vehicle Type  
'''  
  
df\_SOC = df.groupby(['OAT\_Category', 'Vehicle Type'])[  
 ['HV Battery Power[Watts]', 'Air Conditioning Power[Watts]', 'Heater Power[Watts]','HV Battery SOC[%]','FCR']  
].mean().reset\_index().sort\_values(by='HV Battery SOC[%]', ascending=False)  
  
df\_SOC

OAT\_Category Vehicle Type HV Battery Power[Watts] \  
4 Cool EV -6315.025345   
12 Mild EV -4864.866869   
0 Cold EV -7207.394049   
18 Warm PHEV -4609.834834   
11 Hot PHEV -3295.594514   
3 Cold PHEV -3308.952767   
9 Extremely Cold PHEV -1156.463762   
15 Mild PHEV -3634.517430   
7 Cool PHEV -3375.550045   
1 Cold HEV NaN   
2 Cold ICE NaN   
5 Cool HEV NaN   
6 Cool ICE NaN   
8 Extremely Cold ICE NaN   
10 Hot ICE NaN   
13 Mild HEV NaN   
14 Mild ICE NaN   
16 Warm HEV NaN   
17 Warm ICE NaN   
  
 Air Conditioning Power[Watts] Heater Power[Watts] HV Battery SOC[%] \  
4 407.628054 401.791590 78.880645   
12 64.395251 141.293594 78.480394   
0 524.328587 1493.045840 72.497332   
18 818.198198 NaN 56.076689   
11 747.712418 NaN 42.988390   
3 4.091151 NaN 28.827266   
9 0.000000 NaN 27.144109   
15 218.946951 NaN 26.340167   
7 24.847430 NaN 25.019190   
1 NaN NaN NaN   
2 NaN NaN NaN   
5 NaN NaN NaN   
6 NaN NaN NaN   
8 NaN NaN NaN   
10 NaN NaN NaN   
13 NaN NaN NaN   
14 NaN NaN NaN   
16 NaN NaN NaN   
17 NaN NaN NaN   
  
 FCR   
4 NaN   
12 NaN   
0 NaN   
18 NaN   
11 NaN   
3 1.588181   
9 5.838999   
15 0.676201   
7 1.106321   
1 0.549312   
2 0.859137   
5 0.560842   
6 0.862787   
8 0.846985   
10 0.696799   
13 0.573515   
14 0.853879   
16 0.193009   
17 1.015872

'''  
Data exploration of Battery Power, Fuel Consumption Rate (FCR) using the features below with respect to Trip and Vehicle Type  
'''  
  
df\_EC\_trip = df.groupby(['Trip','Vehicle Type'])[  
 ['Latitude[deg]',  
 'Longitude[deg]',  
 'Air Conditioning Power[Watts]',  
 'Heater Power[Watts]',  
 'Vehicle Speed[km/h]',  
 'Distance[km]',  
 'Engine RPM[RPM]',  
 'OAT[DegC]',  
 'Generalized\_Weight',  
 'FCR',  
 'HV Battery Power[Watts]',  
 'MAF[g/sec]',  
 'Absolute Load[%]',  
 'Short Term Fuel Trim Bank 1[%]',  
 'Short Term Fuel Trim Bank 2[%]',  
 'Long Term Fuel Trim Bank 1[%]',  
 'Long Term Fuel Trim Bank 2[%]'  
 ]  
].mean().reset\_index().sort\_values(by=['FCR','HV Battery Power[Watts]'], ascending=False)  
  
df\_EC\_trip

Trip Vehicle Type Latitude[deg] Longitude[deg] \  
737 422 PHEV 42.287141 -83.725580   
503 315 PHEV 42.267723 -83.719078   
1049 544 PHEV 42.295950 -83.702473   
3723 1565 PHEV 42.294440 -83.792688   
620 368 PHEV 42.281735 -83.718567   
... ... ... ... ...   
4994 2670 ICE 42.230204 -83.698985   
5034 2883 ICE 42.276690 -83.681240   
5035 2889 ICE 42.242079 -83.680689   
5036 2898 ICE 42.272055 -83.685443   
5039 2928 ICE 42.254532 -83.682352   
  
 Air Conditioning Power[Watts] Heater Power[Watts] Vehicle Speed[km/h] \  
737 NaN NaN 60.795111   
503 0.000000 NaN 42.921597   
1049 NaN NaN 59.629204   
3723 NaN NaN 65.545201   
620 0.125392 NaN 38.494302   
... ... ... ...   
4994 NaN NaN 68.260870   
5034 NaN NaN 51.305556   
5035 NaN NaN 40.365625   
5036 NaN NaN 43.899899   
5039 NaN NaN 23.826667   
  
 Distance[km] Engine RPM[RPM] OAT[DegC] Generalized\_Weight FCR \  
737 0.576187 2334.161290 1.500000 4000.000000 8.519646   
503 2.772324 504.244681 8.915780 4000.000000 8.167473   
1049 1.348090 2188.994924 -3.010152 4000.000000 8.029910   
3723 8.649201 2108.468750 -18.709821 4000.000000 7.875599   
620 9.066935 786.875829 -6.256781 3038.577456 7.782446   
... ... ... ... ... ...   
4994 1.517008 1953.159420 NaN 3500.000000 NaN   
5034 0.746649 1213.638889 7.046296 3500.000000 NaN   
5035 1.750702 1223.231250 5.400000 3500.000000 NaN   
5036 5.995511 1215.314459 3.338726 3500.000000 NaN   
5039 0.404235 1019.646667 9.386667 3500.000000 NaN   
  
 HV Battery Power[Watts] MAF[g/sec] Absolute Load[%] \  
737 5723.351390 NaN NaN   
503 -4192.833443 0.710000 NaN   
1049 3143.135527 NaN NaN   
3723 7965.864873 NaN NaN   
620 276.822704 6.275787 NaN   
... ... ... ...   
4994 NaN NaN NaN   
5034 NaN 14.180926 27.567175   
5035 NaN 15.231781 30.715687   
5036 NaN 15.655702 32.528401   
5039 NaN 10.829733 27.100654   
  
 Short Term Fuel Trim Bank 1[%] Short Term Fuel Trim Bank 2[%] \  
737 NaN NaN   
503 NaN NaN   
1049 NaN NaN   
3723 NaN NaN   
620 NaN NaN   
... ... ...   
4994 3.430707 NaN   
5034 -1.121238 -9.194155   
5035 -1.086426 -9.291992   
5036 -0.201435 -9.765230   
5039 -0.192708 -10.145833   
  
 Long Term Fuel Trim Bank 1[%] Long Term Fuel Trim Bank 2[%]   
737 NaN NaN   
503 NaN NaN   
1049 NaN NaN   
3723 NaN NaN   
620 NaN NaN   
... ... ...   
4994 3.509964 NaN   
5034 NaN NaN   
5035 NaN NaN   
5036 NaN NaN   
5039 NaN NaN   
  
[5050 rows x 19 columns]

'''  
Data exploration of Battery Power, Fuel Consumption Rate (FCR) using the features below with respect to Date (Month) and Vehicle Type  
'''  
  
df\_EC\_time = df.groupby([df['Date'].dt.to\_period('M'), 'Vehicle Type'])[  
 ['Latitude[deg]',  
 'Longitude[deg]',  
 'Air Conditioning Power[Watts]',  
 'Heater Power[Watts]',  
 'Vehicle Speed[km/h]',  
 'Distance[km]',  
 'Engine RPM[RPM]',  
 'OAT[DegC]',  
 'Generalized\_Weight',  
 'FCR',  
 'HV Battery Power[Watts]',  
 'MAF[g/sec]',  
 'Absolute Load[%]',  
 'Short Term Fuel Trim Bank 1[%]',  
 'Short Term Fuel Trim Bank 2[%]',  
 'Long Term Fuel Trim Bank 1[%]',  
 'Long Term Fuel Trim Bank 2[%]']  
].mean().reset\_index().sort\_values(  
 by=['Date', 'FCR', 'HV Battery Power[Watts]'],  
)  
  
df\_EC\_time

Date Vehicle Type Latitude[deg] Longitude[deg] \  
1 2017-11 HEV 42.270885 -83.730286   
2 2017-11 ICE 42.271662 -83.728795   
3 2017-11 PHEV 42.276155 -83.724590   
0 2017-11 EV 42.270438 -83.729372   
5 2017-12 HEV 42.270560 -83.727416   
6 2017-12 ICE 42.270985 -83.727754   
7 2017-12 PHEV 42.272422 -83.724337   
4 2017-12 EV 42.275912 -83.729516   
9 2018-01 HEV 42.272277 -83.731813   
10 2018-01 ICE 42.270453 -83.730027   
11 2018-01 PHEV 42.271369 -83.726888   
8 2018-01 EV 42.271260 -83.733565   
13 2018-02 HEV 42.272391 -83.730405   
14 2018-02 ICE 42.269730 -83.728094   
15 2018-02 PHEV 42.275730 -83.725041   
12 2018-02 EV 42.278501 -83.725578   
17 2018-03 HEV 42.272847 -83.730845   
18 2018-03 ICE 42.270907 -83.727818   
19 2018-03 PHEV 42.275792 -83.724699   
16 2018-03 EV 42.277992 -83.732349   
21 2018-04 HEV 42.269748 -83.725555   
22 2018-04 ICE 42.268554 -83.735553   
23 2018-04 PHEV 42.276921 -83.722950   
20 2018-04 EV 42.272069 -83.737230   
  
 Air Conditioning Power[Watts] Heater Power[Watts] Vehicle Speed[km/h] \  
1 NaN NaN 43.413799   
2 NaN NaN 37.604944   
3 107.837861 NaN 40.055765   
0 385.080724 387.570512 37.845037   
5 NaN NaN 41.474029   
6 NaN NaN 36.485543   
7 8.106811 NaN 38.090622   
4 527.820062 725.497236 38.324684   
9 NaN NaN 41.960510   
10 NaN NaN 37.654817   
11 17.582774 NaN 40.156629   
8 455.500315 1454.074057 31.269648   
13 NaN NaN 42.520909   
14 NaN NaN 36.634668   
15 19.820653 NaN 38.768772   
12 392.385820 1257.371732 38.245497   
17 NaN NaN 45.173367   
18 NaN NaN 39.407010   
19 23.728703 NaN 41.802540   
16 443.470375 306.493349 38.568744   
21 NaN NaN 42.573222   
22 NaN NaN 41.171133   
23 0.331506 NaN 41.765474   
20 61.290323 0.000000 18.419892   
  
 Distance[km] Engine RPM[RPM] OAT[DegC] Generalized\_Weight FCR \  
1 4.422087 1053.625865 NaN 3213.726498 0.542582   
2 4.351819 1384.739340 NaN 3553.919165 0.876889   
3 6.130402 446.089312 5.302876 3606.249474 1.122664   
0 5.197760 NaN 4.947384 3500.000000 NaN   
5 5.158407 1091.908296 NaN 3230.512406 0.539653   
6 4.932551 1370.856595 NaN 3541.763087 0.818075   
7 6.039241 550.119223 -1.480307 3637.054524 1.276928   
4 4.999694 NaN -0.503159 3500.000000 NaN   
9 4.753501 1160.532423 1.865404 3203.124223 0.564790   
10 4.324497 1406.320488 2.255533 3480.709759 0.843201   
11 6.023829 650.278231 -2.448563 3614.626963 1.731219   
8 3.850865 NaN -2.189553 3500.000000 NaN   
13 5.070865 1094.106371 -0.948706 3213.218805 0.550496   
14 4.095413 1379.465206 -1.193078 3510.487505 0.817684   
15 5.126571 558.951124 -1.642965 3673.636777 1.419964   
12 4.065962 NaN -2.063397 3500.000000 NaN   
17 4.899186 1090.729829 3.046485 3192.742259 0.560943   
18 4.177044 1413.343291 3.433087 3479.413527 0.834096   
19 5.553451 507.260776 3.717426 3651.435940 1.061479   
16 4.590405 NaN 2.556929 3500.000000 NaN   
21 5.845298 1036.902053 4.706287 3260.138221 0.544288   
22 4.463937 1467.693137 6.505307 3355.574546 0.845495   
23 4.104111 548.225346 4.067274 3541.263672 1.024962   
20 0.230644 NaN 7.000000 3500.000000 NaN   
  
 HV Battery Power[Watts] MAF[g/sec] Absolute Load[%] \  
1 NaN 7.956748 26.756422   
2 NaN 12.390124 30.489777   
3 -3639.575620 4.288507 NaN   
0 -5819.770797 NaN NaN   
5 NaN 7.960814 26.734022   
6 NaN 11.525376 30.296919   
7 -3245.157058 5.206778 28.515582   
4 -6548.884092 NaN NaN   
9 NaN 8.364218 29.259980   
10 NaN 11.967993 31.730827   
11 -2783.988406 5.721529 0.000000   
8 -6834.245613 NaN NaN   
13 NaN 8.112337 27.038388   
14 NaN 11.514108 30.589958   
15 -3541.478268 5.118278 NaN   
12 -7558.073422 NaN NaN   
17 NaN 8.260457 28.371134   
18 NaN 11.757754 30.868834   
19 -3684.578395 4.929784 NaN   
16 -6354.509141 NaN NaN   
21 NaN 8.022511 25.713681   
22 NaN 12.049428 31.863793   
23 -2913.129046 4.908048 NaN   
20 -3317.303763 NaN NaN   
  
 Short Term Fuel Trim Bank 1[%] Short Term Fuel Trim Bank 2[%] \  
1 -0.672727 0.048961   
2 0.372626 -0.100306   
3 NaN NaN   
0 NaN NaN   
5 -0.812959 -0.377534   
6 0.432193 -0.373544   
7 0.624510 NaN   
4 NaN NaN   
9 -0.683583 -0.122051   
10 0.340978 0.109015   
11 0.000000 NaN   
8 NaN NaN   
13 -0.488588 -0.270337   
14 0.440682 -0.158966   
15 NaN NaN   
12 NaN NaN   
17 -0.595930 0.669341   
18 0.406216 -0.055941   
19 NaN NaN   
16 NaN NaN   
21 -0.422664 -0.876713   
22 0.288203 0.069593   
23 NaN NaN   
20 NaN NaN   
  
 Long Term Fuel Trim Bank 1[%] Long Term Fuel Trim Bank 2[%]   
1 0.118748 -0.704893   
2 1.424386 1.556238   
3 NaN NaN   
0 NaN NaN   
5 -0.383901 -1.230228   
6 1.733919 1.184792   
7 -2.265380 NaN   
4 NaN NaN   
9 -0.709057 -1.111838   
10 1.030568 1.145690   
11 -2.343750 NaN   
8 NaN NaN   
13 -0.279954 -1.663008   
14 1.370126 1.323453   
15 NaN NaN   
12 NaN NaN   
17 -0.037885 -1.371812   
18 1.691747 1.471730   
19 NaN NaN   
16 NaN NaN   
21 1.303382 -1.672551   
22 1.333101 1.122821   
23 NaN NaN   
20 NaN NaN

'''  
Data exploration of Battery Power, Fuel Consumption Rate (FCR) using the features below with respect to Date and Vehicle Type  
'''  
  
df\_EC\_time = df.groupby([df['Date'].dt.to\_period('D'), 'Vehicle Type'])[  
 ['Latitude[deg]',  
 'Longitude[deg]',  
 'Air Conditioning Power[Watts]',  
 'Heater Power[Watts]',  
 'Vehicle Speed[km/h]',  
 'Distance[km]',  
 'Engine RPM[RPM]',  
 'OAT[DegC]',  
 'Generalized\_Weight',  
 'FCR',  
 'HV Battery Power[Watts]',  
 'MAF[g/sec]',  
 'Absolute Load[%]',  
 'Short Term Fuel Trim Bank 1[%]',  
 'Short Term Fuel Trim Bank 2[%]',  
 'Long Term Fuel Trim Bank 1[%]',  
 'Long Term Fuel Trim Bank 2[%]']  
].mean().reset\_index().sort\_values(  
 by=['Date', 'FCR', 'HV Battery Power[Watts]'],  
)  
  
df\_EC\_time

Date Vehicle Type Latitude[deg] Longitude[deg] \  
3 2017-11-01 PHEV 42.278371 -83.731898   
0 2017-11-01 EV 42.278134 -83.757882   
1 2017-11-01 HEV 42.271430 -83.735358   
2 2017-11-01 ICE 42.275332 -83.735707   
7 2017-11-02 PHEV 42.267811 -83.725619   
.. ... ... ... ...   
561 2018-04-02 ICE 42.266908 -83.740012   
562 2018-04-02 PHEV 42.274750 -83.723151   
565 2018-04-03 PHEV 42.280450 -83.726182   
563 2018-04-03 HEV 42.288396 -83.737577   
564 2018-04-03 ICE 42.273095 -83.733424   
  
 Air Conditioning Power[Watts] Heater Power[Watts] Vehicle Speed[km/h] \  
3 423.254438 NaN 34.632641   
0 58.928571 2089.285714 34.581428   
1 NaN NaN 44.235172   
2 NaN NaN 32.259101   
7 47.353144 NaN 38.296344   
.. ... ... ...   
561 NaN NaN 41.533589   
562 0.809717 NaN 44.604928   
565 NaN NaN 36.928471   
563 NaN NaN 46.990847   
564 NaN NaN 37.696869   
  
 Distance[km] Engine RPM[RPM] OAT[DegC] Generalized\_Weight FCR \  
3 2.675273 147.130922 8.466505 3724.401426 0.000000   
0 0.503022 NaN 5.000000 3500.000000 NaN   
1 3.635668 1014.955255 NaN 3199.366191 NaN   
2 3.874759 1320.427124 NaN 3540.393564 NaN   
7 6.834930 486.426222 4.270162 3444.920027 0.000000   
.. ... ... ... ... ...   
561 5.763351 1398.940764 7.849522 3483.278665 0.907605   
562 3.000267 706.920571 5.170571 3272.571429 3.659103   
565 2.899305 0.000000 4.621512 3774.527453 0.000000   
563 6.099540 1066.613096 2.361616 3282.872734 0.595142   
564 4.151050 1466.746329 5.492239 3070.238095 0.747004   
  
 HV Battery Power[Watts] MAF[g/sec] Absolute Load[%] \  
3 -5329.249002 3.163527 NaN   
0 -4338.209821 NaN NaN   
1 NaN 7.562402 25.968902   
2 NaN 11.650380 29.370518   
7 -2857.820257 4.426551 NaN   
.. ... ... ...   
561 NaN 12.660379 31.731291   
562 -935.198050 5.319922 NaN   
565 -8162.208439 NaN NaN   
563 NaN 8.901670 27.338117   
564 NaN 10.240529 29.443318   
  
 Short Term Fuel Trim Bank 1[%] Short Term Fuel Trim Bank 2[%] \  
3 NaN NaN   
0 NaN NaN   
1 NaN NaN   
2 NaN NaN   
7 NaN NaN   
.. ... ...   
561 -0.228289 0.060634   
562 NaN NaN   
565 NaN NaN   
563 -0.695988 -2.955995   
564 0.050979 -0.822193   
  
 Long Term Fuel Trim Bank 1[%] Long Term Fuel Trim Bank 2[%]   
3 NaN NaN   
0 NaN NaN   
1 NaN NaN   
2 NaN NaN   
7 NaN NaN   
.. ... ...   
561 1.276508 0.879580   
562 NaN NaN   
565 NaN NaN   
563 0.375059 -1.382212   
564 3.058880 4.153914   
  
[566 rows x 19 columns]

'''  
Data exploration of Battery Power, Fuel Consumption Rate (FCR) using the features Speed, Absolute Load[%], Engine RPM, OAT, Generalized\_Weight with respect to Latitude, Longitude and Vehicle Type  
'''  
  
df\_la\_lo = df.groupby(['Latitude[deg]','Longitude[deg]','Vehicle Type'])[  
 ['Vehicle Speed[km/h]', 'Absolute Load[%]','Engine RPM[RPM]','OAT[DegC]','Generalized\_Weight','FCR','HV Battery Power[Watts]']  
].mean().reset\_index().sort\_values(by=['FCR','HV Battery Power[Watts]'], ascending=False)  
  
df\_la\_lo

Latitude[deg] Longitude[deg] Vehicle Type Vehicle Speed[km/h] \  
231563 42.243933 -83.734674 ICE 51.000000   
242871 42.244170 -83.718042 ICE 52.000000   
243350 42.244181 -83.717473 ICE 60.000000   
245129 42.244224 -83.716918 ICE 60.000000   
240874 42.244129 -83.718571 ICE 46.333333   
... ... ... ... ...   
1856114 42.325111 -83.748894 ICE 80.000000   
1856131 42.325151 -83.748901 ICE 85.500000   
1856189 42.325349 -83.748998 ICE 97.857143   
1856193 42.325380 -83.748911 ICE 90.833333   
1856212 42.325568 -83.802335 ICE 71.333333   
  
 Absolute Load[%] Engine RPM[RPM] OAT[DegC] Generalized\_Weight \  
231563 NaN 2037.000000 3.0 6500.0   
242871 NaN 1658.500000 3.0 6500.0   
243350 NaN 1774.000000 3.0 6500.0   
245129 NaN 1308.500000 3.0 6500.0   
240874 NaN 1424.333333 3.0 6500.0   
... ... ... ... ...   
1856114 NaN 1367.800000 NaN 5000.0   
1856131 74.509804 1689.000000 NaN 4500.0   
1856189 NaN 1640.000000 NaN 5000.0   
1856193 NaN 1527.666667 NaN 5000.0   
1856212 16.470589 2714.333333 NaN 3000.0   
  
 FCR HV Battery Power[Watts]   
231563 49.700001 NaN   
242871 25.250000 NaN   
243350 24.350000 NaN   
245129 24.350000 NaN   
240874 20.716667 NaN   
... ... ...   
1856114 NaN NaN   
1856131 NaN NaN   
1856189 NaN NaN   
1856193 NaN NaN   
1856212 NaN NaN   
  
[1856229 rows x 10 columns]

'''  
ICE, HEV, EV and PHEV Analysis (Distance vs FCR and HV Battery Power[Watts]  
'''  
  
df\_EC\_trip\_ICE\_HEV = df\_EC\_trip[df\_EC\_trip['Vehicle Type'].isin(['ICE','HEV'])]  
df\_EC\_trip\_EV\_PHEV = df\_EC\_trip[df\_EC\_trip['Vehicle Type'].isin(['EV','PHEV'])]  
  
df\_EC\_time\_ICE\_HEV = df\_EC\_time[df\_EC\_time['Vehicle Type'].isin(['ICE','HEV'])]  
df\_EC\_time\_EV\_PHEV = df\_EC\_time[df\_EC\_time['Vehicle Type'].isin(['EV','PHEV'])]

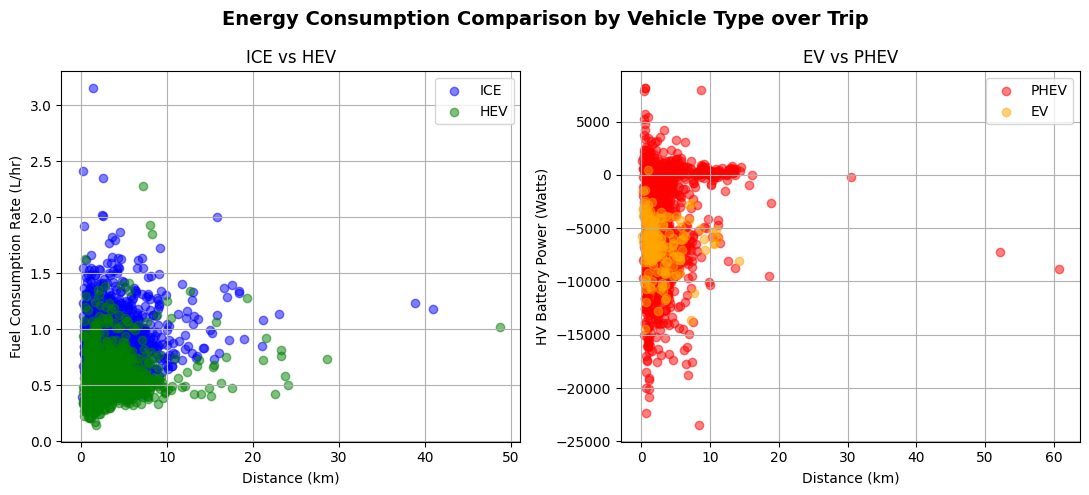
for i in df\_EC\_trip\_ICE\_HEV.columns:  
 print(f"Unique values in column '{i}':")  
 print(df\_EC\_trip\_ICE\_HEV[i].unique())  
 print(df\_EC\_trip\_ICE\_HEV[i].value\_counts())

Unique values in column 'Trip':  
[2628 2645 2648 ... 2889 2898 2928]  
Trip  
107 2  
639 2  
160 2  
340 2  
1800 2  
 ..  
2549 1  
2546 1  
2542 1  
2531 1  
2519 1  
Name: count, Length: 2316, dtype: int64  
Unique values in column 'Vehicle Type':  
['ICE' 'HEV']  
Vehicle Type  
ICE 2200  
HEV 1586  
Name: count, dtype: int64  
Unique values in column 'Latitude[deg]':  
[42.29362649 42.27365438 42.25314284 ... 42.24207904 42.27205511  
 42.25453229]  
Latitude[deg]  
42.254532 1  
42.293626 1  
42.273654 1  
42.253143 1  
42.305868 1  
 ..  
42.256033 1  
42.275071 1  
42.259447 1  
42.278615 1  
42.298313 1  
Name: count, Length: 3786, dtype: int64  
Unique values in column 'Longitude[deg]':  
[-83.73190393 -83.74155843 -83.73935583 ... -83.68068876 -83.68544342  
 -83.68235187]  
Longitude[deg]  
-83.682352 1  
-83.731904 1  
-83.741558 1  
-83.739356 1  
-83.734312 1  
 ..  
-83.712491 1  
-83.741150 1  
-83.739479 1  
-83.739980 1  
-83.727332 1  
Name: count, Length: 3786, dtype: int64  
Unique values in column 'Air Conditioning Power[Watts]':  
[nan]  
Series([], Name: count, dtype: int64)  
Unique values in column 'Heater Power[Watts]':  
[nan]  
Series([], Name: count, dtype: int64)  
Unique values in column 'Vehicle Speed[km/h]':  
[44.1 24.26388889 36.55291577 ... 40.365625 43.89989889  
 23.82666667]  
Vehicle Speed[km/h]  
38.520792 2  
40.500000 2  
46.428571 2  
30.005076 1  
45.045161 1  
 ..  
88.485106 1  
47.000000 1  
27.669911 1  
88.109244 1  
96.187342 1  
Name: count, Length: 3783, dtype: int64  
Unique values in column 'Distance[km]':  
[1.37466111 0.25470486 2.55578642 ... 1.750702 5.99551064 0.404235 ]  
Distance[km]  
0.404235 1  
1.374661 1  
0.254705 1  
2.555786 1  
7.217842 1  
 ..  
9.207089 1  
3.217346 1  
3.998192 1  
3.591849 1  
8.299708 1  
Name: count, Length: 3786, dtype: int64  
Unique values in column 'Engine RPM[RPM]':  
[1552.06875 1287.88888889 1373.21814255 ... 1223.23125 1215.31445905  
 1019.64666667]  
Engine RPM[RPM]  
1019.646667 1  
1552.068750 1  
1287.888889 1  
1373.218143 1  
1742.126582 1  
 ..  
2793.723608 1  
1370.185185 1  
1157.881902 1  
1226.991667 1  
1507.251969 1  
Name: count, Length: 3786, dtype: int64  
Unique values in column 'OAT[DegC]':  
[ nan 8.61265823 -7.04639175 ... 5.4 3.33872599  
 9.38666667]  
OAT[DegC]  
 0.000000 49  
 3.000000 37  
 4.000000 34  
 7.000000 30  
-2.000000 25  
 ..  
 1.393382 1  
 1.080000 1  
 6.514019 1  
 5.104854 1  
 7.852174 1  
Name: count, Length: 1709, dtype: int64  
Unique values in column 'Generalized\_Weight':  
[ nan 4000. 3911.61290323 ... 3319.37799043 4001.75746924  
 3929.9719888 ]  
Generalized\_Weight  
3000.000000 536  
3500.000000 379  
4000.000000 130  
2500.000000 127  
4500.000000 78  
 ...   
4653.191489 1  
3348.591549 1  
3303.398058 1  
3516.953573 1  
3929.971989 1  
Name: count, Length: 2405, dtype: int64  
Unique values in column 'FCR':  
[3.15467445 2.41160611 2.35345339 ... 0.16966979 0.14629346 nan]  
FCR  
0.146293 1  
3.154674 1  
2.411606 1  
2.353453 1  
2.281641 1  
 ..  
1.636976 1  
1.639703 1  
1.649497 1  
1.660195 1  
1.724061 1  
Name: count, Length: 3582, dtype: int64  
Unique values in column 'HV Battery Power[Watts]':  
[nan]  
Series([], Name: count, dtype: int64)  
Unique values in column 'MAF[g/sec]':  
[43.10675009 33.90625 32.63060481 ... 15.23178096 15.65570238  
 10.82973306]  
MAF[g/sec]  
10.829733 1  
43.106750 1  
33.906250 1  
32.630605 1  
32.731645 1  
 ..  
24.259259 1  
23.819149 1  
24.383926 1  
23.740803 1  
24.803167 1  
Name: count, Length: 3613, dtype: int64  
Unique values in column 'Absolute Load[%]':  
[37.93382408 31.91721167 31.24380682 ... 30.71568688 32.52840113  
 27.10065394]  
Absolute Load[%]  
27.100654 1  
37.933824 1  
31.917212 1  
31.243807 1  
45.731448 1  
 ..  
41.940624 1  
40.549374 1  
41.715687 1  
43.798127 1  
41.377057 1  
Name: count, Length: 3677, dtype: int64  
Unique values in column 'Short Term Fuel Trim Bank 1[%]':  
[-0.32714844 -1.82291667 -0.15186285 ... -1.08642578 -0.20143453  
 -0.19270833]  
Short Term Fuel Trim Bank 1[%]  
 0.173611 3  
-0.683594 3  
-0.781250 3  
-1.121238 3  
 0.000000 3  
 ..  
-1.054734 1  
 0.264164 1  
 0.088778 1  
-0.595238 1  
 0.177317 1  
Name: count, Length: 3676, dtype: int64  
Unique values in column 'Short Term Fuel Trim Bank 2[%]':  
[ -0.15136719 0.09765625 0.1737986 ... -9.29199219 -9.76523003  
 -10.14583333]  
Short Term Fuel Trim Bank 2[%]  
 0.000000 8  
 0.390625 3  
-0.407609 2  
 0.321691 2  
-0.781250 2  
 ..  
 0.291667 1  
-1.242188 1  
-1.311678 1  
-2.937500 1  
-0.151367 1  
Name: count, Length: 1711, dtype: int64  
Unique values in column 'Long Term Fuel Trim Bank 1[%]':  
[5.91796875 5.36024306 5.88890389 ... 0.32249273 0.49308402 3.50996377]  
Long Term Fuel Trim Bank 1[%]  
 0.000000 19  
-0.781250 9  
-2.343750 9  
-1.562500 8  
 3.906250 8  
 ..  
 0.168429 1  
 4.708150 1  
-2.271412 1  
 3.838453 1  
 5.360243 1  
Name: count, Length: 3606, dtype: int64  
Unique values in column 'Long Term Fuel Trim Bank 2[%]':  
[ 5.26855469 4.21006944 4.10704644 ... -1.14583333 -0.24786421  
 -3.17195012]  
Long Term Fuel Trim Bank 2[%]  
 0.000000 21  
 0.781250 9  
 3.125000 9  
-4.687500 7  
 1.562500 6  
 ..  
 1.522550 1  
-0.670662 1  
-0.493260 1  
 2.560996 1  
 4.164043 1  
Name: count, Length: 1624, dtype: int64

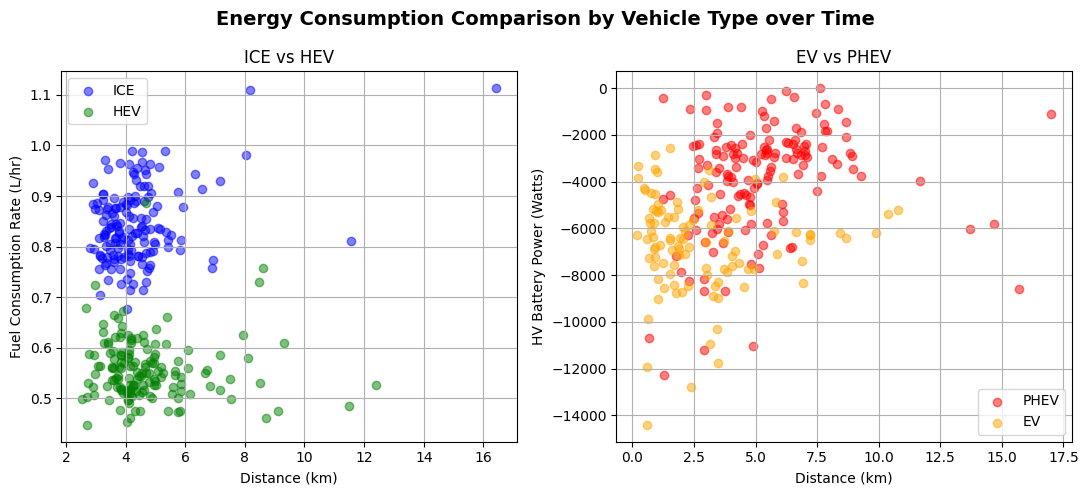
for i in df\_EC\_trip\_EV\_PHEV.columns:  
 print(f"Unique values in column '{i}':")  
 print(df\_EC\_trip\_EV\_PHEV[i].unique())  
 print(df\_EC\_trip\_EV\_PHEV[i].value\_counts())

Unique values in column 'Trip':  
[ 422 315 544 ... 1651 706 1779]  
Trip  
575 2  
625 2  
1198 2  
1009 2  
1615 2  
 ..  
779 1  
1022 1  
948 1  
596 1  
1565 1  
Name: count, Length: 1174, dtype: int64  
Unique values in column 'Vehicle Type':  
['PHEV' 'EV']  
Vehicle Type  
PHEV 1072  
EV 192  
Name: count, dtype: int64  
Unique values in column 'Latitude[deg]':  
[42.28714052 42.26772324 42.29594973 ... 42.28599205 42.23178769  
 42.26242699]  
Latitude[deg]  
42.253724 1  
42.264457 1  
42.250996 1  
42.256739 1  
42.263710 1  
 ..  
42.281735 1  
42.294440 1  
42.295950 1  
42.267723 1  
42.287141 1  
Name: count, Length: 1264, dtype: int64  
Unique values in column 'Longitude[deg]':  
[-83.72558023 -83.71907758 -83.70247277 ... -83.75801275 -83.75688737  
 -83.69414626]  
Longitude[deg]  
-83.739821 1  
-83.797908 1  
-83.798650 1  
-83.739577 1  
-83.691512 1  
 ..  
-83.718567 1  
-83.792688 1  
-83.702473 1  
-83.719078 1  
-83.725580 1  
Name: count, Length: 1264, dtype: int64  
Unique values in column 'Air Conditioning Power[Watts]':  
[ nan 0.00000000e+00 1.25391850e-01 8.57114625e+02  
 5.34883721e+00 7.94117647e+01 1.93873704e+02 1.34730539e+01  
 4.38756856e-01 9.22260274e+01 4.74747475e+01 1.60774818e+01  
 3.02631579e+01 4.38865918e+01 6.73366834e+01 8.73328088e+00  
 8.36092715e+01 6.03495146e+01 9.49111470e+01 1.33056133e+00  
 6.37049704e+02 1.62219399e+01 7.81402679e+01 1.02083333e+02  
 2.06683047e+02 1.58730159e+01 3.95780591e+01 4.08510638e+01  
 1.08843537e+01 1.02094241e+02 5.26315789e+01 4.36363636e+01  
 1.23247126e+03 1.79439252e+00 2.40963855e+01 1.85448944e+03  
 2.44938272e+02 3.79821958e+00 1.84232365e+01 1.15744337e+03  
 3.58208955e+01 8.04123711e+01 1.26631702e+02 9.22580645e+01  
 7.53926702e+00 2.26237425e+02 1.35384615e+02 2.67123288e+01  
 5.96231494e+01 1.30491132e+02 5.42076503e+01 3.89243876e+01  
 5.50083822e+01 1.89792551e+02 2.42696629e+01 8.91588785e+01  
 1.82325581e+02 1.38396624e+02 2.12198221e+01 2.66105630e+01  
 2.18072289e+01 1.65289256e+00 2.02865330e+02 2.49453978e+02  
 5.24604569e+01 1.11184211e+02 2.46680498e+01 7.70234987e+00  
 2.89855072e+00 1.07899461e+02 5.95523013e+02 4.17952314e+01  
 5.17129356e+01 3.92647059e+01 8.27123696e+01 2.89516567e+01  
 9.51244460e+00 3.98836458e+01 7.56022674e+01 4.11639723e+02  
 1.55586987e+01 8.54363535e+01 5.58613295e+01 4.59016393e+01  
 3.61965192e+02 7.77027027e+01 1.24027073e+02 6.93877551e+01  
 9.20156047e+01 1.56453716e+01 5.49324721e+01 1.21435143e+01  
 6.03821656e+01 7.64462810e+00 4.05816259e+01 7.41029641e+00  
 6.12265416e+01 4.66324786e+01 7.29585007e+00 2.23436706e+01  
 2.24321608e+01 2.12041885e+01 3.87481371e+00 2.01024209e+02  
 1.83440000e+02 1.47879177e+02 1.12108317e+02 1.90034762e+01  
 8.66666667e+01 1.71854200e+02 1.35000000e+02 1.19727891e+02  
 3.41476539e+02 9.41634241e+01 1.78508772e+02 5.06843267e+01  
 1.40327869e+01 3.59434629e+02 1.28775357e+02 2.64864865e+01  
 4.71176471e+02 1.16475300e+02 9.67582418e+02 4.68485450e+02  
 9.06049822e+02 3.31683168e+01 7.31389365e+01 1.93928571e+02  
 2.60483871e+02 6.05160550e+02 3.29461615e+02 1.15000000e+03  
 2.76816609e+00 1.07542579e+01 6.47347347e+02 3.85825826e+01  
 3.15835962e+02 1.05263158e+01 1.23402062e+02 1.50000000e+02  
 7.44623656e+01 4.86956522e+00 3.71031746e+02 7.74774775e+00  
 6.12903226e+01 7.57812500e+02 7.89733465e-01 8.33333333e+01  
 9.23076923e+01 3.50000000e+02 1.05000000e+03 4.79878049e+02  
 1.14899713e+02 7.63157895e+02 3.63086233e+00 2.50191939e+02  
 3.90073529e+02 6.48648649e+01 3.85546039e+02 1.04700599e+03  
 5.22222222e+02 3.16591928e+02 2.85678392e+02 3.41697192e+02  
 4.50000000e+02 2.09037559e+02 2.17582418e+02 5.89285714e+01  
 9.54674221e+01 6.98113208e+01 1.85365854e+01 1.02384738e+01  
 4.95192308e+01 8.41463415e+02 7.16413793e+02 2.90909091e+02  
 3.51342282e+02 8.75711382e+02 5.85774059e+00 6.53034301e+01  
 2.98468849e+02 7.58394161e+01 4.08757062e+02 9.04180064e+01  
 1.16666667e+02 3.50290698e+01 2.65682657e+01 3.97652418e+02  
 6.94868586e+02 2.16541353e+01 7.99492386e+02 3.08050847e+02  
 3.25609756e+02 7.86200717e+02 2.59562842e+01 2.71037464e+02  
 2.65716180e+02 9.82201405e+01 2.53472222e+01 2.36848635e+02  
 4.00000000e+02 7.05370370e+02 3.78668942e+02 3.96308017e+02  
 3.49282297e+02 8.98795181e+02 6.76465201e+02 3.82550336e+01  
 5.09302326e+01 5.02792793e+02 3.76811594e+01 5.17317073e+02  
 1.10000000e+03 4.29978166e+02 1.06310680e+03 4.46288365e+02  
 7.75347913e+01 5.14162113e+02 3.89267016e+02 2.73536585e+02  
 1.14485294e+03 1.49336942e+02 4.98746867e+02 2.58364780e+02  
 4.01215805e+01 5.29680365e+00 1.06991525e+03 6.50155376e+02  
 5.40540541e+00 1.14210526e+03 9.11685824e+02 4.29687500e+01  
 7.80481283e+02 1.21269113e+03 9.80362196e+02 2.10494753e+01  
 3.21037464e+02 1.07251908e+03 2.72222222e+02 6.42252682e+02  
 1.07802850e+03 4.19864560e+01 3.04761905e+01 9.92100840e+02  
 9.00000000e+02 4.10588235e+01 3.38235294e+02 2.10921502e+02  
 9.29965753e+02 6.37641026e+02 1.05483871e+03 7.23051948e+02  
 9.61915888e+02 7.27472527e+02 2.67105719e+02 6.11466165e+02  
 6.52427184e+01 7.50876384e+02 8.79629630e+01 1.11538462e+02  
 9.81697613e+02 9.72607656e+02 1.09927798e+03 1.08028391e+03  
 5.26666667e+02 5.20091324e+02 3.93474265e+02 7.86134454e+02  
 8.12903226e+01 1.87866928e+00 2.51196172e+00 4.05172414e+02  
 1.00343750e+03 1.10727924e+03 2.53122807e+02 1.09975962e+03  
 7.98242188e+02 4.79461279e+01 3.60986547e+01 8.29097840e+02  
 4.05967977e+02 6.35321101e+02 7.24641148e+02 1.82048872e+02  
 3.47683398e+02 9.54089219e+02 7.67275748e+02 9.82093664e+02  
 2.16289593e+01 8.97453704e+02 4.11764706e+01 1.07713675e+03  
 1.94693572e+02 6.76363636e+00 3.75888717e+01 8.74345238e+02  
 6.35416667e+00 2.75283019e+02 4.30029297e+02 9.53044041e+02  
 2.59594384e+01 1.30105900e+01 4.45689655e+02 3.01449275e+01  
 2.74846626e+01 5.01253133e-01 1.96463654e+01 1.11164260e+03  
 3.61993603e+02 1.06317992e+03 8.82870370e+02 1.17647059e+01  
 3.98045603e+02 4.59210526e+02 1.13994911e+01 9.74307958e+02  
 1.20388350e+01 9.44262295e+01 2.71269615e+02 7.36956522e+02  
 3.25531915e+01 2.48349835e+02 7.87323944e+02 3.47455471e+02  
 1.12222222e+02 7.81976744e+02 1.01750000e+03 1.54666667e+01  
 6.04166667e+01 1.77977528e+02 1.53080645e+03 4.28417722e+02  
 1.11278195e+01 1.07400000e+03 1.27872340e+02 7.71604938e+01  
 7.00000000e+02 4.30107527e+00 1.02702703e+02 2.90000000e+02  
 5.98507463e+02 2.86567164e+01 4.92664093e+01 1.17070845e+03  
 1.18238636e+03 4.27636364e+02]  
Air Conditioning Power[Watts]  
0.000000 656  
350.000000 4  
450.000000 2  
400.000000 2  
900.000000 2  
 ...   
16.221940 1  
78.140268 1  
102.083333 1  
206.683047 1  
47.474747 1  
Name: count, Length: 345, dtype: int64  
Unique values in column 'Heater Power[Watts]':  
[ nan 187.5 0. 528.43137255 152.41935484  
 500. 214.96496496 750. 220.23809524 480.46875  
 47.33727811 343.47826087 399.3902439 821.05263158 708.08383234  
 373.93162393 81.99871877 1379.73137973 2089.28571429 910.76487252  
 141.89189189 169.27083333 758.41346154 420.73170732 464.13793103  
 612.41610738 189.02439024 1086.4116095 11.0876452 687.5  
 2226.68810289 201.12123336 627.03379224 625.63451777 547.39467849  
 520.609319 1606.55737705 66.28242075 16.90981432 1315.10416667  
 460.29776675 250. 727.77777778 198.80546075 2337.30715288  
 1060.24096386 283.88278388 2268.45637584 496.3963964 285.36585366  
 1182.24299065 1419.87179487 1071.50655022 664.2394822 346.71663098  
 118.16939891 109.75609756 178.92156863 327.06766917 743.6440678  
 593.38098198 1000. 993.29501916 340.90909091 916.66666667  
 596.63271081 305.55555556 316.44815256 1142.51781473 2576.88284519  
 281.93277311 3017.79661017 1238.53211009 1476.02739726 2040.  
 219.35483871 1126.1682243 542.58241758 3092.83819629 1096.80451128  
 2534.98340708 751.15313653 27.00617284 271.36752137 358.09018568  
 422.84688995 460.28880866 706.98924731 97.61904762 152.96803653  
 873.94957983 3911.16751269 264.36781609 1382.8125 886.03818616  
 859.375 735.3515625 3008.40807175 897.71283355 960.6741573  
 195.77874818 1854.35779817 426.11524164 2478.15764482 599.66777409  
 901.51515152 547.00854701 3264.24501425 2843.94904459 822.91666667  
 447.75390625 827.72020725 3752.90979631 79.74137931 3174.92492492  
 542.8700361 2501.59914712 741.10878661 2625.43706294 379.62962963  
 3549.04632153 2021.62629758 2617.55771567 3147.95918367 2184.81848185  
 2250. 221.89922481 3662.69230769 1250. 2213.48314607  
 1199.19354839 3835.57844691 164.87341772 678.63636364 2536.15520282  
 560.63432836 1118.52861035 1204.54545455]  
Heater Power[Watts]  
0.000000 51  
250.000000 3  
750.000000 2  
1000.000000 2  
500.000000 2  
 ..  
678.636364 1  
2536.155203 1  
560.634328 1  
1118.528610 1  
1204.545455 1  
Name: count, Length: 137, dtype: int64  
Unique values in column 'Vehicle Speed[km/h]':  
[60.79511089 42.92159718 59.62920368 ... 52.14861397 59.97120739  
 90.68342077]  
Vehicle Speed[km/h]  
47.943826 1  
65.918925 1  
70.798292 1  
39.069764 1  
63.285435 1  
 ..  
38.494302 1  
65.545201 1  
59.629204 1  
42.921597 1  
60.795111 1  
Name: count, Length: 1264, dtype: int64  
Unique values in column 'Distance[km]':  
[0.57618744 2.77232383 1.34808963 ... 0.66334484 1.08363414 8.36971529]  
Distance[km]  
2.666954 1  
3.734729 1  
2.340162 1  
3.132310 1  
7.276682 1  
 ..  
9.066935 1  
8.649201 1  
1.348090 1  
2.772324 1  
0.576187 1  
Name: count, Length: 1264, dtype: int64  
Unique values in column 'Engine RPM[RPM]':  
[2.33416129e+03 5.04244681e+02 2.18899492e+03 2.10846875e+03  
 7.86875829e+02 2.04616901e+03 1.98909250e+03 1.88781818e+03  
 2.21453704e+03 7.47740741e+02 2.20887143e+03 2.85786590e+02  
 1.70582006e+03 1.34017629e+03 1.79898182e+03 3.63222727e+02  
 6.60686851e+02 1.59935857e+03 1.74887225e+03 7.59880000e+02  
 8.63676423e+02 1.53412583e+03 6.36259226e+02 1.54669281e+03  
 1.49328986e+03 8.78811465e+02 8.22547067e+02 1.36931313e+03  
 1.46377160e+03 1.46075897e+03 7.80534462e+02 6.21982143e+02  
 1.56315789e+03 5.67904762e+02 1.01354562e+03 1.41109890e+03  
 7.69752358e+02 1.24511314e+03 7.41561275e+02 9.96494877e+02  
 1.20071905e+03 1.28317708e+03 4.22298738e+02 1.11358112e+03  
 7.63380734e+02 1.06318378e+03 1.11515903e+03 1.29371503e+03  
 1.21416977e+03 1.14144681e+03 7.34008171e+02 1.11843155e+03  
 8.42484424e+02 1.21565417e+03 1.19736986e+03 8.71135314e+02  
 1.08540107e+03 1.16883152e+03 9.98966292e+02 1.04388513e+03  
 9.97776699e+02 1.04214790e+03 1.20296057e+03 1.12092055e+03  
 9.18938462e+02 1.07127049e+03 3.42797814e+02 1.01566492e+03  
 9.91868009e+02 9.48312649e+02 5.65424390e+02 3.80394366e+02  
 8.68981132e+02 9.02239645e+02 9.27386364e+02 9.85603175e+02  
 9.57195051e+02 8.71758759e+02 8.70213457e+02 7.20575360e+02  
 8.42094607e+02 8.88347134e+02 8.20682803e+02 7.51434685e+02  
 8.82884786e+02 8.60428571e+02 7.40784694e+02 9.71369427e+02  
 7.93616852e+02 1.39965455e+03 6.40673321e+02 8.28232353e+02  
 7.27910387e+02 4.82343822e+02 9.08955189e+02 7.12734296e+02  
 7.54748235e+02 7.42857405e+02 8.74546667e+02 7.58754088e+02  
 7.83571556e+02 7.20734683e+02 6.67033203e+02 9.73519495e+02  
 8.06586319e+02 6.05817181e+02 2.80063457e+02 7.98483276e+02  
 6.93119984e+02 1.29604819e+03 1.89008377e+02 6.33992874e+02  
 8.17205899e+02 7.94166667e+02 6.71683138e+02 6.00371901e+02  
 5.06463636e+02 5.32046823e+02 3.05864571e+02 5.79407249e+02  
 4.24526012e+02 4.73359023e+02 6.17898927e+02 5.60855003e+02  
 5.18160870e+02 5.28122995e+02 4.69321429e+02 9.97264520e+02  
 4.87268156e+02 5.22146771e+02 4.24421842e+02 6.61821867e+02  
 6.53578051e+02 4.66471204e+02 6.71034615e+02 4.60840404e+02  
 8.17755596e+02 5.09753275e+02 3.57271012e+02 8.11407407e+02  
 4.13982867e+02 3.71225000e+02 3.75345214e+02 4.14010204e+02  
 3.35448223e+02 9.99073944e+02 7.12116883e+02 6.45506066e+02  
 8.90592694e+02 2.46684441e+02 4.88396694e+02 3.36255556e+02  
 2.70117877e+02 9.74547558e+02 3.23498029e+02 4.65192279e+02  
 3.00756911e+02 2.92664316e+02 5.58064343e+02 3.08051601e+02  
 5.99727907e+02 6.84922705e+02 4.37658316e+02 3.05826677e+02  
 3.31509346e+02 2.11233618e+02 2.58480469e+02 5.42084795e+02  
 4.67079602e+02 3.63920240e+02 3.07688633e+02 8.50334006e+02  
 2.26447059e+02 3.81794661e+02 4.97775326e+02 2.03375443e+02  
 2.79355019e+02 2.23879265e+02 2.03027656e+02 4.19452962e+02  
 2.06348774e+02 4.62086792e+02 5.34461145e+02 7.38558109e+02  
 4.11936759e+02 4.96653731e+02 2.10286517e+02 2.97707527e+02  
 3.42572704e+02 1.48753731e+02 3.08493373e+02 8.15483871e+01  
 5.43140271e+02 9.83236010e+01 4.15413105e+01 3.36131007e+02  
 7.06521739e-02 4.94603175e+01 0.00000000e+00 6.76763497e+02  
 6.49079533e+02 5.09244557e+02 7.32319776e+02 5.65459834e+02  
 7.87951220e+02 6.81560000e+02 7.04949631e+02 5.71376855e+02  
 5.62657980e+02 1.13942798e+03 3.24072993e+02 5.97315217e+02  
 7.96512563e+02 5.93775076e+02 9.63584733e+02 4.00914710e+02  
 3.20390698e+02 4.15786492e+02 1.03535646e+03 6.05557252e+02  
 3.51354128e+02 2.69090659e+02 2.24142857e+02 4.31932138e+02  
 4.11094716e+02 2.17674370e+02 2.32669216e+02 2.20932346e+02  
 9.67623888e+01 1.53820982e+02 2.67916331e+02 1.78648649e+02  
 1.98198582e+02 2.95461318e+02 5.04466210e+01 1.24119360e+02  
 3.26146027e+02 2.07603534e+02 7.70481802e+02 5.22839506e+02  
 1.59092702e+01 7.18960177e+01 2.11583514e+01 2.77322799e+02  
 5.35251142e+02 3.98230088e+01 1.27677551e+03 1.84362222e+03  
 6.95956522e+02 1.92365385e+03 7.57256410e+02 9.05052632e+02  
 7.54295652e+02 1.17419718e+03 8.28982063e+02 1.07858667e+03  
 8.66736318e+02 1.03110256e+03 8.19812500e+02 8.60960052e+02  
 8.81173804e+02 1.13300995e+03 7.36017110e+02 8.54966200e+02  
 5.40528302e+02 6.27383041e+02 4.55762611e+02 4.54870968e+02  
 9.94293194e+02 1.27360966e+03 1.01717308e+03 6.72135266e+02  
 1.22952427e+03 9.42835616e+02 8.72656347e+02 6.51480847e+02  
 6.80890411e+02 6.90998654e+02 9.62036212e+02 8.36680761e+02  
 7.51268182e+02 7.45459283e+02 1.00161074e+03 7.71221498e+02  
 6.82043656e+02 8.14907104e+02 1.52372063e+03 1.04730044e+03  
 1.27176513e+03 7.81899399e+02 1.08523256e+03 7.49014909e+02  
 1.33821524e+03 1.24980000e+03 6.20544845e+02 7.78068394e+02  
 7.68445074e+02 7.10211693e+02 4.75491525e+02 8.85084507e+02  
 8.61476510e+02 7.53929213e+02 1.35505140e+03 7.81548173e+02  
 1.14914768e+03 6.86466471e+02 7.70011436e+02 6.33217910e+02  
 8.70906433e+02 9.30549312e+02 6.29647127e+02 8.85388278e+02  
 7.67989157e+02 1.42031373e+03 7.00046512e+02 5.80143166e+02  
 1.02517906e+03 6.97597826e+02 7.24267049e+02 7.74957878e+02  
 6.56367442e+02 7.28363796e+02 7.00288210e+02 1.47156475e+03  
 7.21199226e+02 6.18805195e+02 1.02587220e+03 7.54463911e+02  
 1.11368395e+03 nan 9.55242690e+02 6.36289522e+02  
 7.68197952e+02 6.93435986e+02 7.21686307e+02 7.26657293e+02  
 1.12655102e+03 6.92055483e+02 8.79883402e+02 5.98916264e+02  
 6.26196041e+02 7.45819270e+02 8.21187448e+02 7.17430224e+02  
 7.91175614e+02 6.64103957e+02 7.64597561e+02 7.04977503e+02  
 8.92127660e+02 6.11304706e+02 6.63748235e+02 1.08524257e+03  
 7.52620513e+02 8.36893805e+02 9.91890196e+02 8.80445455e+02  
 7.03552906e+02 7.61140227e+02 6.02626984e+02 8.91997446e+02  
 8.16064059e+02 8.36200978e+02 7.94290829e+02 6.78463478e+02  
 6.41153519e+02 8.82933025e+02 9.21686932e+02 1.09453552e+03  
 8.53415370e+02 7.31755767e+02 7.66360137e+02 8.66050725e+02  
 8.50625641e+02 6.43592328e+02 1.00596442e+03 6.80478261e+02  
 1.12192557e+03 8.58446108e+02 7.74013897e+02 4.50412619e+02  
 9.39829596e+02 7.18375523e+02 8.04309912e+02 8.78203030e+02  
 7.89298731e+02 8.14280585e+02 7.48547081e+02 7.87841315e+02  
 7.38416393e+02 9.00434783e+02 1.44019236e+03 6.77217715e+02  
 9.05839644e+02 9.58220894e+02 6.72863669e+02 5.82162162e+02  
 5.35419628e+02 1.12652650e+03 5.47816014e+02 7.46920000e+02  
 7.20797647e+02 5.55326673e+02 7.02075251e+02 6.51676531e+02  
 5.51391325e+02 7.29593968e+02 6.61501986e+02 7.19347204e+02  
 7.45869855e+02 1.30444872e+03 7.45480671e+02 7.38361799e+02  
 6.53552901e+02 7.07862595e+02 7.72838086e+02 6.96122424e+02  
 7.35481953e+02 7.20239669e+02 7.40720025e+02 9.64652850e+02  
 7.17680106e+02 9.96650026e+02 5.84724652e+02 5.99693939e+02  
 7.02976744e+02 7.09941315e+02 6.02987265e+02 8.50710280e+02  
 6.87274444e+02 1.02165203e+03 7.05972892e+02 9.70478632e+02  
 7.52306878e+02 1.20448676e+03 1.18539759e+03 7.44108434e+02  
 8.66068249e+02 7.94030675e+02 7.74172650e+02 8.36072700e+02  
 8.09294118e+02 7.45661515e+02 9.11859031e+02 6.97244221e+02  
 9.82284703e+02 7.21619256e+02 8.83317073e+02 1.00813036e+03  
 1.34922283e+03 1.00174146e+03 9.57046135e+02 5.20473348e+02  
 8.63718499e+02 7.16292512e+02 3.38630137e+02 6.90753927e+02  
 6.50503012e+02 1.03329887e+03 1.08215812e+03 2.01559754e+03  
 8.04983607e+02 9.92590909e+02 8.64043956e+02 7.56450394e+02  
 4.83550351e+02 9.95026022e+02 1.29032636e+03 1.03543162e+03  
 9.08882682e+02 6.38605634e+02 1.24621036e+03 1.04946400e+03  
 8.84556553e+02 1.24124739e+03 7.42640884e+02 8.46036254e+02  
 8.27663239e+02 7.34568765e+02 4.05988395e+02 9.04687651e+02  
 8.72064399e+02 8.86067723e+02 1.16164286e+03 7.23780997e+02  
 9.86840336e+02 1.78680519e+03 8.60996475e+02 5.39972727e+02  
 9.23131783e+02 6.13107527e+02 9.10824089e+02 6.91412655e+02  
 7.61428571e+02 7.68138272e+02 8.57522505e+02 9.31296491e+02  
 5.97392034e+02 8.66056075e+02 7.17015692e+02 3.43275801e+02  
 9.01372807e+02 6.46134146e+02 9.01660777e+02 6.30027211e+02  
 9.55107817e+02 5.34184211e+02 9.03948187e+02 9.95754098e+02  
 1.24388856e+03 9.80873984e+02 8.30827676e+02 3.40077922e+02  
 6.58358621e+02 6.55609126e+02 9.95276265e+02 9.78980702e+02  
 1.17677473e+03 6.85331288e+02 8.30771552e+02 9.01087302e+02  
 7.77857426e+02 6.18753086e+02 9.33169978e+02 1.04022295e+03  
 9.60297872e+02 7.65937807e+02 6.17200707e+02 7.63808858e+02  
 4.91852742e+02 8.76853392e+02 8.39660633e+02 1.11633088e+03  
 9.26266160e+02 8.92324324e+02 7.50912809e+02 7.55198052e+02  
 7.61727637e+02 1.17889796e+03 5.61822034e+02 3.36000000e+03  
 1.03609655e+03 1.85096774e+02 1.25340377e+03 4.68935847e+02  
 8.37997863e+02 2.99126005e+02 1.67236000e+03 5.25626168e+02  
 2.92128655e+02 1.16030110e+03 4.91798680e+02 8.42087081e+02  
 6.43227503e+02 8.91406518e+02 1.15823214e+03 6.43149897e+02  
 7.67743440e+02 8.07259516e+02 5.90507418e+02 1.28178920e+03  
 5.70771290e+02 5.11893293e+02 7.52184985e+02 8.69772871e+02  
 4.05661765e+02 2.58141388e+02 1.39125475e+01 6.03027778e+02  
 1.82571930e+02 3.60828179e+02 1.10919337e+03 3.98921459e+02  
 4.33177419e+02 6.12590038e+02 2.11153153e+02 6.37193581e+02  
 7.27353406e+02 5.28125283e+02 8.81601942e+02 6.35859613e+02  
 4.43501272e+02 6.61051437e+02 1.81337838e+02 5.04016064e+02  
 5.70773006e+02 6.19347193e+02 4.36357588e+02 9.18638211e+02  
 4.85116057e+02 1.66835681e+02 7.16967930e+02 6.06486486e+01  
 2.90767162e+02 6.34440167e+02 5.97828467e+02 8.45862069e+02  
 4.06531357e+02 7.16313008e+02 1.00027333e+03 5.17884379e+02  
 8.11366883e+02 2.99225926e+02 1.08904592e+03 9.83626761e+01  
 1.55008568e+03 1.71273138e+02 9.91820128e+02 4.37463700e+02  
 7.83011834e+02 1.18502591e+02 8.28537037e+02 1.69173278e+02  
 2.76478049e+02 7.71772071e+02 1.64146429e+02 4.03173145e+02  
 1.66671512e+02 7.48695652e+01 4.13480334e+02 3.25177665e+02  
 5.22867133e+02 6.81581056e+02 3.85055266e+02 3.98109422e+02  
 9.79452055e+01 9.62005348e+02 5.27443255e+02 6.72113879e+02  
 3.66376404e+02 3.37394231e+02 2.70082737e+02 8.56692708e+01  
 4.34304410e+02 1.57284047e+02 5.77151424e+01 1.77890339e+02  
 3.68484536e+02 3.29058201e+02 1.10938420e+03 1.00064865e+03  
 6.74662045e+01 3.87372340e+02 4.72015656e+01 3.83440922e+02  
 3.18230672e+02 1.28393574e+02 2.65113861e+02 4.67288525e+02  
 1.07350345e+03 7.49076923e+02 8.39965217e+02 5.73260563e+02  
 1.88073930e+02 3.15160075e+02 6.66923077e+01 1.10183072e+03  
 3.55579186e+02 1.11123810e+03 5.10363636e+02 1.70161818e+02  
 2.55295209e+02 8.03545244e+02 6.82648649e+02 9.65183400e+02  
 4.68538700e+02 3.24440336e+02 3.46102964e+02 6.91217391e+02  
 5.57073620e+02 6.43578947e+02 3.68357564e+02 1.23196706e+03  
 6.65420959e+02 3.30868805e+02 2.97643154e+02 2.43987013e+02  
 6.99111111e+02 9.36251920e+01 4.09158249e+02 6.06548544e+02  
 1.24954217e+03 4.51614604e+02 3.24010336e+02 1.55446429e+02  
 3.03914563e+02 1.43709924e+02 2.21044728e+02 6.78581395e+02  
 9.83471125e+02 6.24137705e+02 3.34776735e+02 3.17700645e+02  
 8.07975460e+00 3.05565863e+02 5.29893899e+02 3.26408240e+02  
 2.83769653e+02 4.47751351e+02 1.33172269e+02 1.99559055e+02  
 2.35177215e+02 2.71750000e+02 1.41123967e+02 3.77835681e+02  
 8.59053381e+02 6.37442478e+02 6.91393939e+02 8.80197674e+02  
 1.49043956e+02 3.75172043e+02 2.61239264e+02 1.97900302e+02  
 9.16888889e+01 3.34825503e+02 6.02211765e+01 2.21959732e+02  
 1.62245487e+02 1.40230958e+02 1.37211940e+03 2.00120482e+02  
 1.20618557e+02]  
Engine RPM[RPM]  
0.000000 353  
2334.161290 1  
504.244681 1  
2188.994924 1  
2108.468750 1  
 ...   
162.245487 1  
140.230958 1  
1372.119403 1  
200.120482 1  
120.618557 1  
Name: count, Length: 720, dtype: int64  
Unique values in column 'OAT[DegC]':  
[ 1.5 8.91578014 -3.01015228 ... 8.3 1.31868132  
 -3.48247423]  
OAT[DegC]  
 4.000000 14  
 7.000000 11  
 1.000000 10  
 2.000000 7  
 2.500000 7  
 ..  
 3.488038 1  
-5.041553 1  
-10.412353 1  
 3.227941 1  
-18.709821 1  
Name: count, Length: 1142, dtype: int64  
Unique values in column 'Generalized\_Weight':  
[4000. 3038.5774563 3522.48569092 3902.617398 3500.  
 3534.73684211 3045.00450045 3149.04458599 3084.58390177 3158.40386941  
 3325.88454376 3026.96078431 3043.25032765 3049.65430547 3761.00628931  
 3151.15771079 3920.75015124 3085.35178777 3449.79919679 3116.27906977  
 3361.37306677 3768.79699248 3355.09838998 3369.94949495 3443.20297952  
 3700.58708415 3755.52825553 3223.62637363 3448.14984011 3776.01410935  
 3665.90519703 3434.70319635 3726.67542707 3612.98932384 3543.4434968  
 3690.88319088 3526.31578947 3896.21087315 3118.18778726 3899.87244898  
 3845.14925373 3279.63800905 3903.84615385 3504.16107383 3048.75100725  
 3033.92405063 3085.55399719 3181.63830629 3168.23529412 3763.60039565  
 3812.70358306 3902.37226277 3234.47204969 3852.38693467 3202.38095238  
 3142.1875 3809.88372093 3264.76072094 3637.06140351 3687.02290076  
 3850.4587156 3631.87855787 3698.41269841 3618.43790013 3278.36134454  
 3613.7667304 3718.53932584 3604.44444444 3800.50825921 3721.53024911  
 3647.05882353 3549.56689124 3839.1025641 3605.53129549 3865.58044807  
 3910.10028653 3818.83194279 3857.0029383 3828.23871907 3859.68660969  
 3812.78839815 3728.16901408 3692.03910615 3825.44378698 3899.88938053  
 3847.07158351 3762.30769231 3862.53776435 3000. 3600.56657224  
 3053.52881009 3124.23625255 3145.72864322 3364.08106219 3584.41558442  
 3088.67843306 3246.28975265 3256.49350649 3203.37301587 3249.32975871  
 3323.52941176 3343.18766067 3854.88647581 3456.76500509 3476.55068079  
 3198.30508475 3566.97819315 3686.52849741 3480.2955665 3703.19634703  
 3592.69662921]  
Generalized\_Weight  
4000.000000 759  
3500.000000 232  
3000.000000 165  
3522.485691 1  
3902.617398 1  
 ...   
3566.978193 1  
3686.528497 1  
3480.295567 1  
3703.196347 1  
3592.696629 1  
Name: count, Length: 111, dtype: int64  
Unique values in column 'FCR':  
[8.5196458 8.16747272 8.0299102 7.87559886 7.78244581 7.46703416  
 7.20997976 6.81708192 6.81647807 6.66420724 6.54093721 6.24390373  
 6.06589486 6.06490643 6.04980897 6.02031095 5.80459078 5.76055846  
 5.59427243 5.46785135 5.42890904 5.28232641 5.14470948 5.10283939  
 5.07989036 4.98214219 4.98155163 4.95890011 4.92894989 4.88005643  
 4.64632462 4.64113125 4.62936318 4.53757868 4.52025108 4.51853387  
 4.51134513 4.49967051 4.47283199 4.43019952 4.3317097 4.2821998  
 4.1776191 4.15356391 4.00757819 3.96478161 3.96428621 3.95876211  
 3.94112385 3.9024018 3.80451188 3.77873271 3.77449957 3.73994253  
 3.7074506 3.70303359 3.63263788 3.57878118 3.53889639 3.45711077  
 3.40981786 3.40950151 3.39101126 3.38532665 3.37729256 3.33968977  
 3.32753109 3.23815461 3.23574315 3.14135626 3.13624373 3.06998344  
 3.0640971 3.0521764 3.03420113 2.9822975 2.96429608 2.9406282  
 2.91822063 2.89416467 2.86871198 2.8621247 2.84012506 2.7491414  
 2.74247074 2.70273628 2.65272701 2.649177 2.63711643 2.6277386  
 2.58941621 2.57919833 2.56250901 2.4898903 2.46841793 2.46726784  
 2.46679795 2.4165933 2.41335544 2.34934148 2.3452046 2.34185403  
 2.30862373 2.24285881 2.23646803 2.20079943 2.18291405 2.18091222  
 2.17680477 2.16241931 2.15447794 2.14302397 2.1354386 2.08146025  
 2.02536437 2.01157151 2.010243 1.91162931 1.89632951 1.81165671  
 1.81078309 1.80466181 1.77245876 1.74150283 1.71647729 1.68342034  
 1.671878 1.6510317 1.64810827 1.60888343 1.55964043 1.53362224  
 1.45324008 1.38827021 1.35642939 1.34067274 1.29524649 1.29300826  
 1.28854913 1.27844853 1.25278377 1.24277058 1.20488626 1.20417657  
 1.19705856 1.1670411 1.15206511 1.13936008 1.11710726 1.11194088  
 1.09808092 1.08703787 0.99672041 0.9934344 0.9915353 0.98619871  
 0.97835972 0.97041172 0.9295288 0.92815917 0.87105644 0.86686897  
 0.85484209 0.8453109 0.84430876 0.81083066 0.81038109 0.78968344  
 0.78027143 0.77533034 0.77141174 0.76974127 0.76358962 0.75214846  
 0.69694814 0.68212967 0.67988951 0.67613527 0.65718104 0.65289724  
 0.64232264 0.63297634 0.63220769 0.60933844 0.60902086 0.58410897  
 0.56774686 0.53805374 0.52074917 0.51207871 0.50770757 0.43504862  
 0.35327479 0.29515847 0.24383759 0.16248445 0.0471673 0.01347709  
 0. nan]  
FCR  
0.000000 219  
8.167473 1  
8.519646 1  
7.875599 1  
7.782446 1  
 ...   
0.353275 1  
0.243838 1  
0.162484 1  
0.047167 1  
0.013477 1  
Name: count, Length: 199, dtype: int64  
Unique values in column 'HV Battery Power[Watts]':  
[ 5723.35138993 -4192.8334433 3143.13552749 ... -19945.0365826  
 -20814.21465963 -23445.88962918]  
HV Battery Power[Watts]  
-23445.889629 1  
 5723.351390 1  
-4192.833443 1  
 3143.135527 1  
 7965.864873 1  
 ..  
 230.039381 1  
-9477.314348 1  
 1792.492047 1  
-3400.850649 1  
-8658.818934 1  
Name: count, Length: 1263, dtype: int64  
Unique values in column 'MAF[g/sec]':  
[ nan 0.70999998 6.27578669 2.30288997 0.42999998 5.19475815  
 8.78909904 4.81254465 7.01486512 6.52791342 5.78132169 0.95515989  
 6.86621534 4.8946074 5.63426309 7.50140754 1.80366777 6.07378754  
 6.14040993 8.50792138 7.98675326 11.02387889 1.37441745 0.87  
 6.89798839 5.51878527 0.87061856 4.69429465 8.74882017 6.55407299  
 9.23285433 9.80240863 5.66098191 6.11971635 0.89992662 2.15874994  
 2.27510484 5.27565739 6.58896778 2.53762538 6.37286423 6.70474418  
 5.13206113 11.0343448 3.48222021 9.57201921 8.84383419 7.2768173  
 2.88830443 11.23369841 3.85055991 2.02078157 7.99113124 5.25430155  
 4.46259251 4.2081046 3.5379417 9.58674054 2.36957374 6.08725229  
 2.76768286 5.65874585 4.69756943 0.70999998 5.54740775 8.32562777  
 4.37563408 5.42517881 5.32413915 7.88933548 6.78688809 8.84735995  
 10.01530933 9.8710154 13.95931174 3.89591828 6.48916819 11.4929588  
 7.38476779 11.36155608 0.1701 0.17 5.33388388 5.37151018  
 6.74294314 0.17000845 9.60339748 13.20931945 5.80552347 0.17015464  
 8.52551772 6.14052622 7.12194613 4.66405188 2.22809833 2.61344665  
 0.70999998 2.53445218 2.21616274 2.12267091 6.8298896 3.36056971  
 4.04422653 3.60027939 1.41895078 1.88121947 6.02394546 3.23439162  
 7.77510187 5.50372758 0.92281819 1.27414935 0.87190622 3.07913698  
 0.42999998 19.62172382 12.07979567 18.54466609 7.00445642 18.4682688  
 7.42692293 7.89463643 8.18799981 11.62105612 6.12049312 10.5931331  
 9.87686551 7.19557676 6.51527329 6.66990964 7.14438275 9.23424526  
 5.40692 6.65299518 5.24943386 5.31637414 4.48854591 3.92447572  
 9.17049712 12.49812844 8.27947091 7.0580191 9.94402895 8.97345869  
 6.70196582 5.69249989 7.25431492 5.82407793 8.64392735 9.35082434  
 8.05768162 7.39853403 9.94503332 8.67794768 5.68648351 8.80562821  
 15.61690445 7.85835978 14.61538712 6.14157644 11.25204295 6.51609144  
 15.88132977 9.45957247 5.31957238 5.72325374 6.42023374 5.90945296  
 3.58830497 8.19046939 7.88342264 6.21705602 13.25981277 6.24018592  
 8.83744705 5.91069987 6.40433275 6.29452228 6.89928349 7.11050444  
 5.3756122 7.4129229 6.50480711 14.10858357 8.34733435 5.16567247  
 10.90134963 6.17803127 5.94123771 6.12187193 6.45462771 5.73544803  
 7.03731863 10.44021563 6.44043829 6.29974016 7.2821035 6.0979024  
 11.100296 7.92356707 3.99912674 5.77349817 5.72639631 5.07912438  
 6.24618617 12.37210865 3.98586155 6.92460889 4.96519045 4.86450496  
 6.19620573 6.74944757 5.53166187 6.0409826 5.61924972 7.35225595  
 5.43744645 9.78568067 5.09659992 5.484694 12.36346509 6.20857678  
 6.26152955 9.77443114 8.64438161 5.93214592 6.81768396 5.14342849  
 9.10440595 6.76357369 7.08460602 5.00741892 5.50429853 4.90339622  
 10.22357945 5.92036326 8.53573753 5.90987258 5.51430112 6.45635038  
 8.74032591 9.88435879 5.2460727 7.03053973 5.48563997 8.67433637  
 7.80660162 6.39597541 3.29547523 8.50560523 5.5304497 6.70333735  
 7.17610592 6.41619361 6.78978707 6.35319194 6.64202988 8.243967  
 9.82840555 7.46141782 5.51156443 8.19534503 6.51508618 5.47561263  
 4.48839517 4.94620124 6.98970924 6.23640532 4.62294273 5.67110575  
 4.70458532 5.2354263 5.81723457 4.5527692 4.1496751 5.66791091  
 7.96665781 6.24229131 7.86615367 6.00745663 5.88927628 5.62234798  
 5.57914259 5.55792998 5.20116353 7.94301468 5.56502744 5.75435643  
 8.21628654 6.12892253 5.51237116 4.70870268 5.37040897 6.23881766  
 5.79159142 4.65544225 6.94699358 5.91538877 10.70852008 7.86499984  
 10.39970922 8.68306857 10.07294278 11.94204801 5.89350722 8.70031134  
 4.59828208 5.6066419 5.55962366 8.97808799 6.03190096 7.44184122  
 6.20388428 7.77036811 7.57844623 8.98823147 7.43927945 13.86005406  
 8.10341448 6.46112205 5.69266511 6.93868617 6.02601391 3.41630123  
 5.74938465 7.35436732 11.13806114 10.95316213 21.38729596 6.10941865  
 5.88090895 8.09302174 5.58311799 5.1989226 8.88264851 7.12558561  
 9.8145297 7.79748585 6.94698929 6.0714238 10.84719977 7.9738957  
 10.86857121 6.23292802 8.69567958 6.54467853 7.81102545 4.1040618  
 6.67602891 6.46413109 6.59391574 12.3057853 6.36696396 9.94285695  
 16.19883084 8.11546396 5.58527257 8.9178551 6.6699998 7.10935006  
 5.52451156 7.83793639 8.23827146 8.19148713 7.39096475 4.56373157  
 9.40656516 7.18794565 2.83462627 8.66951737 4.63408527 7.9794875  
 5.17266996 7.89242571 5.63124989 8.33412764 9.65957626 10.09146602  
 8.07540629 7.29699725 3.31012981 5.06391369 5.45271189 9.78348221  
 7.16477178 12.42206018 4.78306742 6.39224121 5.15126968 6.50671271  
 5.9260492 8.32854284 11.12857348 8.84686151 6.47790489 4.89773841  
 7.41634016 5.15877524 8.07157531 8.65893644 4.01283079 9.79939138  
 8.29463946 7.44885885 6.19538946 7.60258991 5.94816309 0.17004073  
 5.68010581 40.22999954 7.05951707 2.28892468 7.39245267 0.42999998  
 3.89593244 5.8669016 3.39455757 19.19104959 5.24525688 2.82064323  
 9.97350809 3.1948844 7.74518898 5.05215357 0.17008016 8.39768417  
 5.52741065 5.61634486 0.17410305 6.79978118 8.42799292 6.31065268  
 13.09429276 5.34992687 5.28196634 6.80278063 8.40640356 5.25305135  
 2.85766061 0.85725599 5.57899988 2.33296486 3.98189856 10.62646386  
 3.21928873 4.93110203 6.03992325 2.52776271 6.75391159 6.68835129  
 5.23316212 5.33694167 4.85810773 2.99402028 5.19071092 2.25594591  
 5.2733131 4.9794232 4.28723484 3.55860699 7.6698169 5.22874391  
 1.91910791 7.46743424 1.02770267 3.75451512 3.5235313 5.92410572  
 4.64721474 0.27708474 2.94007974 5.723821 8.75046644 4.79236151  
 5.87316547 2.70775921 0.47619937 7.46433656 1.370757 14.88574019  
 2.39410829 6.51027825 4.82797412 6.38724838 1.15137303 6.13407396  
 1.64400832 0.42949506 3.38936577 0.17062827 7.8613 1.87530353  
 3.29293277 1.7234956 0.17011111 1.41436389 4.53943971 2.86475881  
 3.22958037 6.81293244 2.88143894 5.11942239 1.28433788 9.73507996  
 3.96152026 4.92023119 2.84064601 3.73490377 3.16879866 0.17015625  
 1.3052734 2.94270263 1.67182876 1.26109443 1.73815923 0.77392775  
 2.7941752 2.35206343 0.82101497 7.61324234 6.76008093 1.28473133  
 3.00329781 1.16837571 3.23109502 2.34211656 1.71090356 2.27272271  
 3.79011468 6.7862067 7.56589046 6.78686939 4.51683087 2.14459141  
 2.72092276 1.038733 7.27880861 0.17183575 3.12581442 7.92696127  
 4.35532463 1.86012723 2.58828432 5.24183285 5.41506746 6.83455134  
 3.39894732 2.60894111 2.79283924 0.74364597 4.30473425 3.76595084  
 5.99254372 3.38882112 7.96571749 4.69484891 2.68447029 2.67607185  
 1.89636359 5.57706658 1.24658983 3.45289555 4.36713583 0.87  
 12.01050174 3.61462465 0.87 2.47131778 1.85898805 3.18799995  
 2.33083966 2.09217247 4.5907557 6.96343448 4.04701631 2.51574104  
 2.88407736 0.71782206 2.72892387 3.11625989 2.68900743 2.50499079  
 4.70632423 1.42716383 2.0377165 2.17623413 0.78999996 3.01134609  
 1.99603299 2.74056332 4.67790026 4.67216803 4.6853029 6.11494177  
 1.88445051 3.15225799 2.46576681 1.93237156 1.21005552 3.21513416  
 1.13458821 2.33409391 1.39801441 1.42334148 11.21373118 1.73289154  
 1.64010308]  
MAF[g/sec]  
0.710000 150  
0.870000 33  
0.710000 10  
0.430000 8  
0.710000 7  
 ...   
1.398014 1  
1.423341 1  
11.213731 1  
1.732892 1  
6.275787 1  
Name: count, Length: 606, dtype: int64  
Unique values in column 'Absolute Load[%]':  
[ nan 28.51558232 0. ]  
Absolute Load[%]  
28.515582 1  
0.000000 1  
Name: count, dtype: int64  
Unique values in column 'Short Term Fuel Trim Bank 1[%]':  
[ nan 0.62451019 0. ]  
Short Term Fuel Trim Bank 1[%]  
0.62451 1  
0.00000 1  
Name: count, dtype: int64  
Unique values in column 'Short Term Fuel Trim Bank 2[%]':  
[nan]  
Series([], Name: count, dtype: int64)  
Unique values in column 'Long Term Fuel Trim Bank 1[%]':  
[ nan -2.26538009 -2.34375 ]  
Long Term Fuel Trim Bank 1[%]  
-2.26538 1  
-2.34375 1  
Name: count, dtype: int64  
Unique values in column 'Long Term Fuel Trim Bank 2[%]':  
[nan]  
Series([], Name: count, dtype: int64)

'''  
Created Scatter plots to show Energy Consumption Comparison by Vehicle Type by Trip  
Distance vs FCR for ICE and HEV  
Distance vs HV Battery Power[Watts] for EV and PHEV  
'''  
  
import matplotlib.pyplot as plt  
  
# Create a 1x2 subplot  
fig, axes = plt.subplots(1, 2, figsize=(11, 5), sharex=False)  
  
# ---------- Subplot 1: ICE vs HEV ----------  
ax1 = axes[0]  
ax1.scatter(  
 df\_EC\_trip\_ICE\_HEV[df\_EC\_trip\_ICE\_HEV['Vehicle Type'] == 'ICE']['Distance[km]'],  
 df\_EC\_trip\_ICE\_HEV[df\_EC\_trip\_ICE\_HEV['Vehicle Type'] == 'ICE']['FCR'],  
 alpha=0.5,  
 c='blue',  
 label='ICE'  
)  
ax1.scatter(  
 df\_EC\_trip\_ICE\_HEV[df\_EC\_trip\_ICE\_HEV['Vehicle Type'] == 'HEV']['Distance[km]'],  
 df\_EC\_trip\_ICE\_HEV[df\_EC\_trip\_ICE\_HEV['Vehicle Type'] == 'HEV']['FCR'],  
 alpha=0.5,  
 c='green',  
 label='HEV'  
)  
ax1.set\_xlabel('Distance (km)')  
ax1.set\_ylabel('Fuel Consumption Rate (L/hr)')  
ax1.set\_title('ICE vs HEV')  
ax1.legend()  
ax1.grid(True)  
  
# ---------- Subplot 2: EV vs PHEV ----------  
ax2 = axes[1]  
ax2.scatter(  
 df\_EC\_trip\_EV\_PHEV[df\_EC\_trip\_EV\_PHEV['Vehicle Type'] == 'PHEV']['Distance[km]'],  
 df\_EC\_trip\_EV\_PHEV[df\_EC\_trip\_EV\_PHEV['Vehicle Type'] == 'PHEV']['HV Battery Power[Watts]'],  
 alpha=0.5,  
 c='red',  
 label='PHEV'  
)  
ax2.scatter(  
 df\_EC\_trip\_EV\_PHEV[df\_EC\_trip\_EV\_PHEV['Vehicle Type'] == 'EV']['Distance[km]'],  
 df\_EC\_trip\_EV\_PHEV[df\_EC\_trip\_EV\_PHEV['Vehicle Type'] == 'EV']['HV Battery Power[Watts]'],  
 alpha=0.5,  
 c='orange',  
 label='EV'  
)  
ax2.set\_xlabel('Distance (km)')  
ax2.set\_ylabel('HV Battery Power (Watts)')  
ax2.set\_title('EV vs PHEV')  
ax2.legend()  
ax2.grid(True)  
  
# ---------- Shared Title ----------  
fig.suptitle('Energy Consumption Comparison by Vehicle Type over Trip', fontsize=14, fontweight='bold')  
plt.tight\_layout(rect=[0, 0, 1, 1]) # Leave space for suptitle  
plt.show()



'''  
Created Scatter plots to show Energy Consumption Comparison by Vehicle Type over Time  
Distance vs FCR for ICE and HEV  
Distance vs HV Battery Power[Watts] for EV and PHEV  
'''  
  
import matplotlib.pyplot as plt  
  
# Create a 1x2 subplot  
fig, axes = plt.subplots(1, 2, figsize=(11, 5), sharex=False)  
  
# ---------- Subplot 1: ICE vs HEV ----------  
ax1 = axes[0]  
ax1.scatter(  
 df\_EC\_time\_ICE\_HEV[df\_EC\_time\_ICE\_HEV['Vehicle Type'] == 'ICE']['Distance[km]'],  
 df\_EC\_time\_ICE\_HEV[df\_EC\_time\_ICE\_HEV['Vehicle Type'] == 'ICE']['FCR'],  
 alpha=0.5,  
 c='blue',  
 label='ICE'  
)  
ax1.scatter(  
 df\_EC\_time\_ICE\_HEV[df\_EC\_time\_ICE\_HEV['Vehicle Type'] == 'HEV']['Distance[km]'],  
 df\_EC\_time\_ICE\_HEV[df\_EC\_time\_ICE\_HEV['Vehicle Type'] == 'HEV']['FCR'],  
 alpha=0.5,  
 c='green',  
 label='HEV'  
)  
ax1.set\_xlabel('Distance (km)')  
ax1.set\_ylabel('Fuel Consumption Rate (L/hr)')  
ax1.set\_title('ICE vs HEV')  
ax1.legend()  
ax1.grid(True)  
  
# ---------- Subplot 2: EV vs PHEV ----------  
ax2 = axes[1]  
ax2.scatter(  
 df\_EC\_time\_EV\_PHEV[df\_EC\_time\_EV\_PHEV['Vehicle Type'] == 'PHEV']['Distance[km]'],  
 df\_EC\_time\_EV\_PHEV[df\_EC\_time\_EV\_PHEV['Vehicle Type'] == 'PHEV']['HV Battery Power[Watts]'],  
 alpha=0.5,  
 c='red',  
 label='PHEV'  
)  
ax2.scatter(  
 df\_EC\_time\_EV\_PHEV[df\_EC\_time\_EV\_PHEV['Vehicle Type'] == 'EV']['Distance[km]'],  
 df\_EC\_time\_EV\_PHEV[df\_EC\_time\_EV\_PHEV['Vehicle Type'] == 'EV']['HV Battery Power[Watts]'],  
 alpha=0.5,  
 c='orange',  
 label='EV'  
)  
ax2.set\_xlabel('Distance (km)')  
ax2.set\_ylabel('HV Battery Power (Watts)')  
ax2.set\_title('EV vs PHEV')  
ax2.legend()  
ax2.grid(True)  
  
# ---------- Shared Title ----------  
fig.suptitle('Energy Consumption Comparison by Vehicle Type over Time', fontsize=14, fontweight='bold')  
plt.tight\_layout(rect=[0, 0, 1, 1]) # Leave space for suptitle  
plt.show()



'''  
Individual Vehicle Type analysis by Trip: df\_ICE, df\_HEV, df\_EV, df\_PHEV  
'''  
  
# Run for trip  
df\_ICE = df\_EC\_trip\_ICE\_HEV[df\_EC\_trip\_ICE\_HEV['Vehicle Type'] == 'ICE']  
df\_HEV = df\_EC\_trip\_ICE\_HEV[df\_EC\_trip\_ICE\_HEV['Vehicle Type'] == 'HEV']  
df\_EV = df\_EC\_trip\_EV\_PHEV[df\_EC\_trip\_EV\_PHEV['Vehicle Type'] == 'EV']  
df\_PHEV = df\_EC\_trip\_EV\_PHEV[df\_EC\_trip\_EV\_PHEV['Vehicle Type'] == 'PHEV']

'''  
Individual Vehicle Type analysis by Time: df\_ICE, df\_HEV, df\_EV, df\_PHEV  
  
  
# Run for time  
df\_ICE = df\_EC\_time\_ICE\_HEV[df\_EC\_time\_ICE\_HEV['Vehicle Type'] == 'ICE']  
df\_HEV = df\_EC\_time\_ICE\_HEV[df\_EC\_time\_ICE\_HEV['Vehicle Type'] == 'HEV']  
df\_EV = df\_EC\_time\_EV\_PHEV[df\_EC\_time\_EV\_PHEV['Vehicle Type'] == 'EV']  
df\_PHEV = df\_EC\_time\_EV\_PHEV[df\_EC\_time\_EV\_PHEV['Vehicle Type'] == 'PHEV']  
  
'''

'''  
Handled Missing values as it is required during Regression  
'''  
  
# show the columns with missing values in df\_ICE and df\_HEV  
  
missing\_values = df\_ICE.isnull().sum()  
print("\nMissing values in ICE Vehicles:")  
print(missing\_values[missing\_values > 0])  
  
missing\_values = df\_HEV.isnull().sum()  
print("\nMissing values in HEV Vehicles:")  
print(missing\_values[missing\_values > 0])  
  
missing\_values = df\_EV.isnull().sum()  
print("\nMissing values in EV Vehicles:")  
print(missing\_values[missing\_values > 0])  
  
missing\_values = df\_PHEV.isnull().sum()  
print("\nMissing values in PHEV Vehicles:")  
print(missing\_values[missing\_values > 0])

Missing values in ICE Vehicles:  
Air Conditioning Power[Watts] 2200  
Heater Power[Watts] 2200  
OAT[DegC] 1025  
Generalized\_Weight 105  
FCR 153  
HV Battery Power[Watts] 2200  
MAF[g/sec] 167  
Absolute Load[%] 103  
Short Term Fuel Trim Bank 1[%] 44  
Short Term Fuel Trim Bank 2[%] 690  
Long Term Fuel Trim Bank 1[%] 51  
Long Term Fuel Trim Bank 2[%] 707  
dtype: int64  
  
Missing values in HEV Vehicles:  
Air Conditioning Power[Watts] 1586  
Heater Power[Watts] 1586  
OAT[DegC] 599  
FCR 51  
HV Battery Power[Watts] 1586  
MAF[g/sec] 6  
Absolute Load[%] 6  
Short Term Fuel Trim Bank 1[%] 36  
Short Term Fuel Trim Bank 2[%] 1368  
Long Term Fuel Trim Bank 1[%] 45  
Long Term Fuel Trim Bank 2[%] 1392  
dtype: int64  
  
Missing values in EV Vehicles:  
Engine RPM[RPM] 192  
FCR 192  
MAF[g/sec] 192  
Absolute Load[%] 192  
Short Term Fuel Trim Bank 1[%] 192  
Short Term Fuel Trim Bank 2[%] 192  
Long Term Fuel Trim Bank 1[%] 192  
Long Term Fuel Trim Bank 2[%] 192  
dtype: int64  
  
Missing values in PHEV Vehicles:  
Air Conditioning Power[Watts] 258  
Heater Power[Watts] 1072  
OAT[DegC] 1  
FCR 655  
HV Battery Power[Watts] 1  
MAF[g/sec] 257  
Absolute Load[%] 1070  
Short Term Fuel Trim Bank 1[%] 1070  
Short Term Fuel Trim Bank 2[%] 1072  
Long Term Fuel Trim Bank 1[%] 1070  
Long Term Fuel Trim Bank 2[%] 1072  
dtype: int64

df\_ICE.describe()

Trip Latitude[deg] Longitude[deg] \  
count 2200.000000 2200.000000 2200.000000   
mean 1194.488636 42.270459 -83.729158   
std 689.257253 0.012903 0.021687   
min 5.000000 42.228171 -83.800113   
25% 616.750000 42.262612 -83.740633   
50% 1167.500000 42.270864 -83.728508   
75% 1731.250000 42.278191 -83.716456   
max 2928.000000 42.316908 -83.679590   
  
 Air Conditioning Power[Watts] Heater Power[Watts] \  
count 0.0 0.0   
mean NaN NaN   
std NaN NaN   
min NaN NaN   
25% NaN NaN   
50% NaN NaN   
75% NaN NaN   
max NaN NaN   
  
 Vehicle Speed[km/h] Distance[km] Engine RPM[RPM] OAT[DegC] \  
count 2200.000000 2200.000000 2200.000000 1175.000000   
mean 38.513763 3.544566 1415.909895 1.676151   
std 10.335123 2.679785 194.666395 6.077748   
min 0.986688 0.059206 650.157400 -30.611236   
25% 32.668991 2.136476 1307.395549 -2.000000   
50% 37.639794 3.036812 1394.249014 1.497778   
75% 43.273612 4.259150 1495.901316 5.255294   
max 92.988889 40.980174 2793.723608 28.986928   
  
 Generalized\_Weight FCR HV Battery Power[Watts] MAF[g/sec] \  
count 2095.000000 2047.000000 0.0 2033.000000   
mean 3472.837328 0.868315 NaN 12.367785   
std 480.131643 0.222176 NaN 3.401966   
min 2500.000000 0.365122 NaN 5.273529   
25% 3168.519599 0.727863 NaN 10.324110   
50% 3500.000000 0.834765 NaN 11.743765   
75% 3714.145658 0.961126 NaN 13.574757   
max 5500.000000 3.154674 NaN 43.106750   
  
 Absolute Load[%] Short Term Fuel Trim Bank 1[%] \  
count 2097.000000 2156.000000   
mean 31.397375 0.368556   
std 4.454647 1.621927   
min 17.912852 -5.815972   
25% 28.696029 -0.501707   
50% 30.957374 0.108277   
75% 33.667582 0.986104   
max 55.477125 13.543159   
  
 Short Term Fuel Trim Bank 2[%] Long Term Fuel Trim Bank 1[%] \  
count 1510.000000 2149.000000   
mean -0.252839 1.463076   
std 3.676007 3.343251   
min -77.315070 -13.831522   
25% -1.100772 -0.376688   
50% -0.333328 1.352447   
75% 0.403827 3.248159   
max 25.615986 26.718750   
  
 Long Term Fuel Trim Bank 2[%]   
count 1493.000000   
mean 1.240369   
std 3.720074   
min -10.548753   
25% -1.168705   
50% 0.975177   
75% 3.426966   
max 20.027608

df\_HEV.describe()

Trip Latitude[deg] Longitude[deg] \  
count 1586.000000 1586.000000 1586.000000   
mean 1090.000000 42.271435 -83.734765   
std 609.153997 0.014667 0.026042   
min 8.000000 42.229036 -83.799807   
25% 620.250000 42.262241 -83.750433   
50% 1023.500000 42.273207 -83.731305   
75% 1442.500000 42.281704 -83.715739   
max 3012.000000 42.313205 -83.679651   
  
 Air Conditioning Power[Watts] Heater Power[Watts] \  
count 0.0 0.0   
mean NaN NaN   
std NaN NaN   
min NaN NaN   
25% NaN NaN   
50% NaN NaN   
75% NaN NaN   
max NaN NaN   
  
 Vehicle Speed[km/h] Distance[km] Engine RPM[RPM] OAT[DegC] \  
count 1586.000000 1586.000000 1586.000000 987.000000   
mean 43.516727 3.634765 1084.958824 1.869060   
std 9.671531 2.893226 229.100779 4.949470   
min 10.932011 0.219721 130.172414 -13.918182   
25% 38.085168 1.937783 932.077829 -1.320829   
50% 42.976737 3.167409 1078.741121 1.928058   
75% 48.347727 4.595484 1235.755318 5.199352   
max 96.187342 48.698232 2311.650206 17.283828   
  
 Generalized\_Weight FCR HV Battery Power[Watts] MAF[g/sec] \  
count 1586.000000 1535.000000 0.0 1580.000000   
mean 3241.805101 0.568199 NaN 8.346108   
std 369.860546 0.180528 NaN 2.579728   
min 2000.000000 0.146293 NaN 2.197069   
25% 3000.000000 0.464805 NaN 6.881587   
50% 3151.434960 0.533781 NaN 7.857364   
75% 3480.243613 0.624716 NaN 9.188845   
max 5000.000000 2.281641 NaN 32.731645   
  
 Absolute Load[%] Short Term Fuel Trim Bank 1[%] \  
count 1580.000000 1550.000000   
mean 27.441848 -0.665138   
std 7.993945 1.191142   
min 3.995943 -7.801827   
25% 23.848746 -1.181007   
50% 26.945488 -0.570376   
75% 30.106458 -0.080356   
max 147.530296 8.016304   
  
 Short Term Fuel Trim Bank 2[%] Long Term Fuel Trim Bank 1[%] \  
count 218.000000 1541.000000   
mean 0.014865 -0.270709   
std 1.586888 2.048271   
min -8.290480 -8.780185   
25% -0.589859 -1.615162   
50% -0.017066 -0.317004   
75% 0.549279 0.939150   
max 4.535953 10.465228   
  
 Long Term Fuel Trim Bank 2[%]   
count 194.000000   
mean -1.529164   
std 2.079918   
min -7.181490   
25% -2.996934   
50% -1.493183   
75% -0.395683   
max 4.164043

df\_PHEV.describe()

Trip Latitude[deg] Longitude[deg] \  
count 1072.000000 1072.000000 1072.000000   
mean 1068.349813 42.273535 -83.724487   
std 595.376940 0.018781 0.024099   
min 2.000000 42.229924 -83.799773   
25% 591.750000 42.259781 -83.739367   
50% 1027.500000 42.277576 -83.721872   
75% 1505.250000 42.285169 -83.708786   
max 2497.000000 42.318748 -83.677901   
  
 Air Conditioning Power[Watts] Heater Power[Watts] \  
count 814.000000 0.0   
mean 31.458010 NaN   
std 132.029855 NaN   
min 0.000000 NaN   
25% 0.000000 NaN   
50% 0.000000 NaN   
75% 0.000000 NaN   
max 1854.489437 NaN   
  
 Vehicle Speed[km/h] Distance[km] Engine RPM[RPM] OAT[DegC] \  
count 1072.000000 1072.000000 1072.000000 1071.000000   
mean 42.361808 3.771616 480.276407 1.181418   
std 11.850411 3.752166 460.683247 6.448809   
min 11.558719 0.037676 0.000000 -20.871176   
25% 35.242796 1.734695 0.000000 -2.789850   
50% 41.526763 3.064417 465.831741 2.016129   
75% 47.297805 4.449652 791.785924 5.460185   
max 101.218062 60.774807 3360.000000 26.222477   
  
 Generalized\_Weight FCR HV Battery Power[Watts] MAF[g/sec] \  
count 1072.000000 417.000000 1071.000000 815.000000   
mean 3778.348617 1.300838 -4719.160889 4.607863   
std 382.654582 1.901255 5010.223152 3.781798   
min 3000.000000 0.000000 -23445.889629 0.170000   
25% 3635.765692 0.000000 -8077.407714 0.870000   
50% 4000.000000 0.000000 -4522.882512 4.708703   
75% 4000.000000 2.236468 -152.108369 6.801281   
max 4000.000000 8.519646 8125.363238 40.230000   
  
 Absolute Load[%] Short Term Fuel Trim Bank 1[%] \  
count 2.000000 2.000000   
mean 14.257791 0.312255   
std 20.163562 0.441595   
min 0.000000 0.000000   
25% 7.128896 0.156128   
50% 14.257791 0.312255   
75% 21.386687 0.468383   
max 28.515582 0.624510   
  
 Short Term Fuel Trim Bank 2[%] Long Term Fuel Trim Bank 1[%] \  
count 0.0 2.000000   
mean NaN -2.304565   
std NaN 0.055416   
min NaN -2.343750   
25% NaN -2.324158   
50% NaN -2.304565   
75% NaN -2.284973   
max NaN -2.265380   
  
 Long Term Fuel Trim Bank 2[%]   
count 0.0   
mean NaN   
std NaN   
min NaN   
25% NaN   
50% NaN   
75% NaN   
max NaN

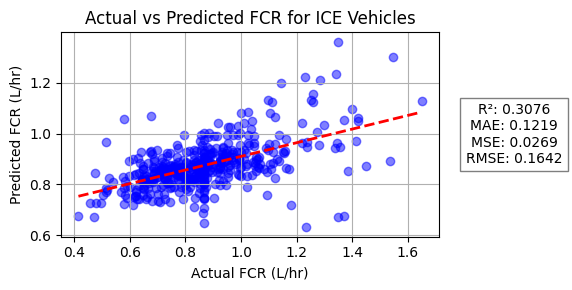
# Handle missing values  
  
df\_ICE.fillna({  
 'OAT[DegC]': 15,  
 'Generalized\_Weight': df\_ICE['Generalized\_Weight'].mean(),  
 'FCR': df\_ICE['FCR'].mean(),  
 'HV Battery Power[Watts]': 0.0, # ICE vehicles typically do not have HV Battery Power  
 'Air Conditioning Power[Watts]' : 0,  
 'Heater Power[Watts]': df\_ICE['Heater Power[Watts]'].mean(),  
 'MAF[g/sec]': df\_ICE['MAF[g/sec]'].mean(),  
 'Absolute Load[%]': df\_ICE['Absolute Load[%]'].mean(),  
 'Short Term Fuel Trim Bank 1[%]': df\_ICE['Short Term Fuel Trim Bank 1[%]'].mean(),  
 'Short Term Fuel Trim Bank 2[%]': df\_ICE['Short Term Fuel Trim Bank 2[%]'].mean(),  
 'Long Term Fuel Trim Bank 1[%]': df\_ICE['Long Term Fuel Trim Bank 1[%]'].mean(),  
 'Long Term Fuel Trim Bank 2[%]': df\_ICE['Long Term Fuel Trim Bank 2[%]'].mean()  
}, inplace=True)  
  
df\_HEV.fillna({  
 'OAT[DegC]': 15,  
 'FCR': df\_HEV['FCR'].mean(),  
 'HV Battery Power[Watts]': 0,  
 'Air Conditioning Power[Watts]': 0,  
 'Heater Power[Watts]': 0,  
 'MAF[g/sec]': df\_HEV['MAF[g/sec]'].mean(),  
 'Absolute Load[%]': df\_HEV['Absolute Load[%]'].mean(),  
 'Short Term Fuel Trim Bank 1[%]': df\_HEV['Short Term Fuel Trim Bank 1[%]'].mean(),  
 'Short Term Fuel Trim Bank 2[%]': df\_HEV['Short Term Fuel Trim Bank 2[%]'].mean(),  
 'Long Term Fuel Trim Bank 1[%]': df\_HEV['Long Term Fuel Trim Bank 1[%]'].mean(),  
 'Long Term Fuel Trim Bank 2[%]': df\_HEV['Long Term Fuel Trim Bank 2[%]'].mean()  
}, inplace=True)  
  
df\_EV.fillna({  
 'Engine RPM[RPM]': 0, # EVs typically do not have engine RPM  
 'FCR': 0, # EVs typically do not have fuel consumption rate   
 'MAF[g/sec]': 0,  
 'Absolute Load[%]': 0,  
 'Short Term Fuel Trim Bank 1[%]': 0,  
 'Short Term Fuel Trim Bank 2[%]': 0,  
 'Long Term Fuel Trim Bank 1[%]': 0,  
 'Long Term Fuel Trim Bank 2[%]': 0  
}, inplace=True)  
  
df\_PHEV.fillna({  
 'OAT[DegC]': 15,  
 'FCR': df\_PHEV['FCR'].mean(),  
 'HV Battery Power[Watts]': 0.0, # PHEVs typically do not have HV Battery Power  
 'Air Conditioning Power[Watts]': 0,  
 'Heater Power[Watts]': 0,  
 'MAF[g/sec]': df\_PHEV['MAF[g/sec]'].mean(),  
 'Absolute Load[%]': df\_PHEV['Absolute Load[%]'].mean(),  
 'Short Term Fuel Trim Bank 1[%]': df\_PHEV['Short Term Fuel Trim Bank 1[%]'].mean(),  
 'Short Term Fuel Trim Bank 2[%]': 0,  
 'Long Term Fuel Trim Bank 1[%]': df\_PHEV['Long Term Fuel Trim Bank 1[%]'].mean(),  
 'Long Term Fuel Trim Bank 2[%]': 0  
}, inplace=True)

"""  
Created a function to Perform linear regression using given features and target.  
Prints model coefficients, regression equation, and evaluation metrics.  
Returns:  
 model: Trained linear regression model.  
 X\_test, y\_test: Test data for further analysis.  
 y\_pred: Predictions on test set.  
 regression\_line\_model: Model for plotting Actual vs Predicted regression line.  
"""  
  
from sklearn.model\_selection import train\_test\_split  
from sklearn.linear\_model import LinearRegression  
from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error  
import numpy as np  
  
def linear\_regression\_analysis(features, target, X, y):  
  
 # Split data  
 X\_train, X\_test, y\_train, y\_test = train\_test\_split(  
 X, y, test\_size=0.2, random\_state=42  
 )  
  
 # Fit linear model  
 model = LinearRegression()  
 model.fit(X\_train, y\_train)  
 y\_pred = model.predict(X\_test)  
  
 # Print model coefficients  
 print("Model Coefficients:")  
 for feature, coef in zip(features, model.coef\_):  
 print(f" {feature}: {coef:.4f}")  
  
 # Print regression equation  
 equation = " + ".join([f"{coef:.4f}\*{feature}" for feature, coef in zip(features, model.coef\_)])  
 print(f"\nRegression Equation:")  
 print(f" Slope of the regression line: {model.coef\_}")  
 print(f" Intercept: {model.intercept\_:.4f}")  
 print(f" Target Variable: {target}")  
 print(f" {target} = {equation} + {model.intercept\_:.4f}\n")  
  
 # Evaluation metrics  
 r2 = r2\_score(y\_test, y\_pred)  
 mae = mean\_absolute\_error(y\_test, y\_pred)  
 mse = mean\_squared\_error(y\_test, y\_pred)  
 rmse = np.sqrt(mse)  
  
 print("Evaluation Metrics:")  
 print(f" R² Score : {r2:.4f}")  
 print(f" MAE : {mae:.4f}")  
 print(f" MSE : {mse:.4f}")  
 print(f" RMSE : {rmse:.4f}\n")  
  
 # Regression line for plotting (optional)  
 regression\_line\_model = LinearRegression()  
 regression\_line\_model.fit(y\_test.values.reshape(-1, 1), y\_pred)  
  
 return model, X\_test, y\_test, y\_pred, regression\_line\_model

# 6 Linear Regression to Predict Energy Consumption in ICE, HEV, EV and PHEV Vehicles

"""  
Scenario 1 for ICE  
Performed linear regression analysis to predict FCR for ICE vehicles using 'Vehicle Speed[km/h]' and 'Distance[km]' as features.  
Trained the model and obtained predictions on the test set.  
Plotted Actual vs Predicted FCR values and the regression line.  
Displayed evaluation metrics (R², MAE, MSE, RMSE) on the plot.  
"""  
  
#features = ['Vehicle Speed[km/h]','Distance[km]', 'Engine RPM[RPM]']  
features = ['Vehicle Speed[km/h]','Distance[km]']  
target = 'FCR'  
X = df\_ICE[features]  
y = df\_ICE[target]  
model, X\_test, y\_test, y\_pred, regression\_line\_model = linear\_regression\_analysis(features, target, X, y)  
  
# plotting the results  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error  
  
y\_test\_sorted = np.sort(y\_test)  
y\_line = regression\_line\_model.predict(y\_test\_sorted.reshape(-1, 1))  
  
plt.figure(figsize=(6, 3))  
plt.scatter(y\_test, y\_pred, alpha=0.5, color='blue')  
# Plotted the regression line  
plt.plot(y\_test\_sorted, y\_line, color='red', linestyle='--', linewidth=2, label='Regression Line')  
plt.title('Actual vs Predicted FCR for ICE Vehicles')  
plt.xlabel('Actual FCR (L/hr)')  
plt.ylabel('Predicted FCR (L/hr)')  
plt.text(1.20, 0.5, f'R²: {r2\_score(y\_test, y\_pred):.4f}\nMAE: {mean\_absolute\_error(y\_test, y\_pred):.4f}\nMSE: {mean\_squared\_error(y\_test, y\_pred):.4f}\nRMSE: {np.sqrt(mean\_squared\_error(y\_test, y\_pred)):.4f}',  
 fontsize=10, ha='center', va='center', transform=plt.gca().transAxes, bbox=dict(facecolor='white', alpha=0.5))  
plt.grid(True)  
plt.tight\_layout()  
plt.show()

Model Coefficients:  
 Vehicle Speed[km/h]: 0.0091  
 Distance[km]: -0.0006  
  
Regression Equation:  
 Slope of the regression line: [ 0.00906548 -0.00058008]  
 Intercept: 0.5228  
 Target Variable: FCR  
 FCR = 0.0091\*Vehicle Speed[km/h] + -0.0006\*Distance[km] + 0.5228  
  
Evaluation Metrics:  
 R² Score : 0.3076  
 MAE : 0.1219  
 MSE : 0.0269  
 RMSE : 0.1642



X = df\_ICE[features]  
print(X.head())  
y = df\_ICE[target]  
print(y.head())

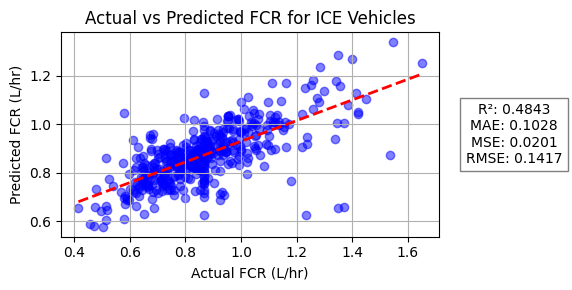
Vehicle Speed[km/h] Distance[km]  
4973 44.100000 1.374661  
4983 24.263889 0.254705  
4986 36.552916 2.555786  
5000 30.005076 2.493146  
3888 45.045161 2.559252  
4973 3.154674  
4983 2.411606  
4986 2.353453  
5000 2.021139  
3888 2.015042  
Name: FCR, dtype: float64

param = [model, X\_test, y\_test, y\_pred, regression\_line\_model, y\_test\_sorted, y\_line]  
  
# print the value of the param variable using for loop  
  
for i, value in enumerate(param):  
 print(f"Parameter {i}: {value}")  
 print() # Add a newline for better readability

Parameter 0: LinearRegression()  
  
Parameter 1: Vehicle Speed[km/h] Distance[km]  
811 36.136471 3.386172  
3176 34.923423 2.114701  
2033 32.450601 2.304397  
74 43.938017 2.014322  
1555 30.492299 4.225037  
... ... ...  
4109 36.339744 2.091850  
442 49.194937 9.278854  
2846 25.014901 0.859650  
283 25.878238 1.149311  
4943 51.938679 2.424305  
  
[440 rows x 2 columns]  
  
Parameter 2: 811 0.745959  
3176 0.771333  
2033 0.673333  
74 0.679211  
1555 0.720144  
 ...   
4109 1.383293  
442 1.345334  
2846 0.613525  
283 0.582152  
4943 0.969332  
Name: FCR, Length: 440, dtype: float64  
  
Parameter 3: [0.84839099 0.83813169 0.81560433 0.91991154 0.79673725 0.86722887  
 0.83994283 0.84380366 0.86159589 0.89090103 0.91910744 0.76802593  
 0.99579399 0.78682127 0.74058429 0.9718213 0.7625944 0.86450701  
 0.86764471 0.84799477 0.97437757 0.82198295 0.76148858 0.89310849  
 0.84865134 0.77656422 0.92055718 0.85088094 0.93394865 0.89132474  
 0.85560155 0.856638 0.92296811 0.90098946 0.80238922 0.85022099  
 0.85544223 0.88030759 0.94443134 0.96258301 0.88552106 0.79619427  
 0.85490548 0.71872201 0.84717766 0.93275236 0.86277992 0.81663092  
 0.8766505 1.15610407 0.86717046 0.85460659 0.8523643 0.86334035  
 0.93557462 0.83356149 1.05639871 0.82038495 0.83123289 0.91502405  
 0.85843618 0.87212636 0.88695056 0.93349629 0.82079547 0.92714164  
 0.89875375 1.03400551 0.85831882 0.90053023 0.87341588 0.81221921  
 0.72659387 0.93176021 0.84771521 0.73317801 0.86033269 0.80766309  
 0.83487138 0.86930155 0.9003098 0.75849599 1.23487705 0.79686298  
 0.7296591 0.82199038 0.64797545 0.9106634 0.87953224 0.8754104  
 0.95647414 1.02534674 0.83413837 0.92528166 0.919599 0.85836506  
 0.99352104 0.88571762 0.88170387 0.9486102 0.75251181 0.66982095  
 0.84246992 0.78510673 0.84517716 0.87168146 0.78853866 0.88068741  
 0.9561485 0.90314197 0.81788266 0.72683313 0.79478895 0.82437551  
 0.87830165 0.86858744 1.1228146 0.78517612 0.78404108 0.94641789  
 0.92567444 0.80311753 0.92886796 0.74332578 0.86400892 0.87544412  
 0.86154108 1.0421233 0.85046371 0.91061435 0.88196565 0.96301728  
 0.88635754 1.13311728 0.92815774 0.85521144 0.86940677 0.89437416  
 0.8196731 0.82377224 1.0164564 0.80407182 0.89121218 0.99247473  
 0.90494684 0.96238605 0.91047694 0.88468909 0.78797995 0.93458729  
 0.88291816 0.74032906 0.84841738 0.94104987 0.8320877 0.90706992  
 0.89481694 0.94994024 1.22104458 0.75645587 0.89809142 0.80482824  
 0.79041853 0.83473093 0.96990176 0.75365069 0.86826434 0.87287877  
 0.90927102 0.76953795 0.89472964 0.9369879 0.82848092 0.88495484  
 0.81744477 0.99219321 0.90838284 0.85257081 0.81865394 0.88573256  
 0.74756805 0.83653008 0.87109021 0.67513341 0.96437306 0.83761795  
 0.91495045 0.90115346 0.83559551 0.76044553 0.8147555 1.0403493  
 0.82453314 0.80279339 0.88795069 0.92142278 0.928968 0.86766642  
 1.00148439 0.85943942 0.83982179 0.78972667 0.81873046 0.74518097  
 1.06258003 0.84919744 0.84631739 0.8100236 0.87440995 0.89258619  
 0.96509886 0.90743151 0.82638332 0.87802414 0.85450882 1.0946267  
 0.95339236 1.06817622 0.81834538 0.89639074 0.87957392 0.8251903  
 0.78234789 0.83916998 0.8213494 0.84737705 1.07951249 0.87209184  
 0.9079723 0.92234277 0.81963978 0.72928722 0.80691006 0.88850128  
 0.823847 0.72326097 0.82054481 0.9346196 0.87343497 0.82852996  
 0.83853255 0.86240996 0.88728663 1.19917882 0.80771779 0.80936051  
 0.92515803 0.90732421 0.81166894 0.86271367 0.75889294 0.89865811  
 0.84127329 1.36206808 0.90303377 0.89445169 0.74594532 0.8613496  
 0.76891289 1.00307922 0.88241166 0.74682413 0.85178095 0.87032263  
 0.93384293 1.20940917 0.89439922 0.91342351 0.92637345 0.82725273  
 0.88929407 0.74230901 0.8769026 0.77137396 0.62987247 0.86033829  
 0.92419427 0.85364777 0.87561948 0.67684774 0.78074333 0.8263613  
 0.81661321 0.84551801 0.8825127 0.93621629 0.83832053 0.74664995  
 0.85127295 0.92907897 0.85514645 0.83230357 0.74371748 0.95910285  
 0.92321336 1.03017344 0.73277594 0.859024 0.79390427 0.81813894  
 0.8612883 0.89461648 0.74981793 1.06524479 0.86080168 0.88191246  
 0.9295701 0.87689083 0.83108736 0.95391923 1.12309396 0.83003957  
 0.95780218 0.87520936 0.86977843 1.08500116 0.87178417 0.85823356  
 0.87036567 1.05045311 0.87195811 0.8386563 1.1278038 0.99015706  
 0.89368108 0.8201755 0.88699138 0.88612876 0.81995987 0.99369952  
 0.79780327 0.82698836 0.89115215 0.85165077 0.84328667 0.85630046  
 0.71164239 0.67096824 0.86459423 0.83620689 0.83374525 0.98901212  
 0.78977676 0.77702194 0.76890408 1.08165265 0.84588445 0.88731165  
 0.86900825 1.00572973 0.80026724 0.94600638 0.90429752 0.92700611  
 1.03982392 0.85596225 0.86158502 0.91945723 1.13205614 0.94154747  
 0.8291596 0.93386854 0.85781837 0.82770528 0.78504621 0.89207723  
 0.71850296 0.80236743 0.78674069 0.96560997 0.92228785 0.81816429  
 0.8518695 0.86072922 1.00227195 0.80400289 0.83246707 0.84416826  
 0.9781007 0.79004734 0.83288358 0.88605025 0.68562347 0.70771792  
 0.81161979 0.79102292 0.77776666 0.82535967 0.80492034 0.88803969  
 0.78784578 0.95282842 0.92328423 0.86796395 0.74250259 0.99371382  
 0.94455087 1.05325916 0.77528562 0.92957879 0.90632815 0.87392105  
 0.82155005 0.83293895 0.87819337 0.83021765 0.78354212 0.82218649  
 0.80917668 0.86471394 1.02457604 0.72552866 0.89843289 0.86840211  
 0.87830259 0.87947578 0.92341055 0.81197158 0.83992783 0.83968368  
 0.94340951 1.30053516 0.7922238 0.94804228 0.91540339 0.89180126  
 1.04770436 0.86016854 0.84562926 0.8272247 0.76476539 0.99037491  
 0.83616515 0.86443375 0.98552575 0.85098457 0.96335402 0.7490342  
 0.75669275 0.99220357]  
  
Parameter 4: LinearRegression()  
  
Parameter 5: [0.41513034 0.45771638 0.46995284 0.47428516 0.4789225 0.49536736  
 0.50329461 0.51374674 0.514379 0.51506285 0.51758335 0.52609581  
 0.53955905 0.56759423 0.57509822 0.5786706 0.57884377 0.57961709  
 0.58215159 0.58418463 0.58891946 0.58943823 0.59715643 0.59826485  
 0.59978136 0.61146053 0.6126709 0.61352484 0.6166284 0.61785171  
 0.62118471 0.62170957 0.6228223 0.62286185 0.62411282 0.62774535  
 0.63121725 0.63286785 0.6329375 0.63580665 0.63872971 0.64316421  
 0.6441142 0.64452313 0.64510878 0.64663727 0.64763717 0.64840961  
 0.64876581 0.64891399 0.65028368 0.65065943 0.65176396 0.65207752  
 0.65739742 0.65915766 0.6630268 0.66348959 0.66413816 0.66437223  
 0.66568546 0.66772863 0.66990758 0.67269108 0.67333332 0.67336171  
 0.67582235 0.67642363 0.67921136 0.67958707 0.68002091 0.68078045  
 0.68244628 0.6831022 0.68334064 0.68529238 0.68641036 0.68670967  
 0.68753643 0.68781848 0.68861174 0.68900049 0.69262187 0.69636851  
 0.69747334 0.70180109 0.70203438 0.7024596 0.70627295 0.7081864  
 0.7087707 0.70924233 0.71193963 0.71450949 0.71558929 0.71602184  
 0.71708106 0.71747095 0.72014362 0.7201471 0.72123459 0.72236614  
 0.72309683 0.72433799 0.72465374 0.72513502 0.7266564 0.72713081  
 0.72758236 0.73044863 0.7321733 0.73261605 0.73290034 0.73302487  
 0.73384182 0.73397081 0.73628736 0.74203561 0.74244517 0.74338945  
 0.74382351 0.74434581 0.74478872 0.74529491 0.74595851 0.74623271  
 0.74632955 0.74738005 0.74861985 0.75679503 0.75746527 0.75748907  
 0.75773521 0.76002507 0.76046094 0.76129776 0.76187781 0.76440163  
 0.76683206 0.76724967 0.76830352 0.76896598 0.77023343 0.77133341  
 0.77279041 0.77361467 0.77395767 0.77401797 0.77591109 0.77652626  
 0.77712971 0.77732579 0.77766622 0.78056469 0.78077904 0.78083438  
 0.7827538 0.78278246 0.78287399 0.78494351 0.78764799 0.79009657  
 0.79091925 0.79192486 0.7923073 0.7960454 0.79848647 0.79873824  
 0.79991969 0.80014764 0.80078638 0.80110845 0.80144661 0.80161938  
 0.80202076 0.80269002 0.80362401 0.80442188 0.80571752 0.81076926  
 0.8118454 0.81398973 0.81672603 0.81716845 0.81750233 0.82003208  
 0.82012972 0.82239168 0.82249976 0.82303097 0.8232893 0.82532131  
 0.82596016 0.82690286 0.82906499 0.8299807 0.82998471 0.83057593  
 0.83235153 0.83308678 0.83426241 0.8347646 0.83493518 0.83804532  
 0.83909954 0.83914393 0.83960617 0.8396761 0.8410291 0.84306891  
 0.84370897 0.8453681 0.84542151 0.84553972 0.84595237 0.84829069  
 0.84858023 0.85042629 0.85108295 0.8511098 0.8520318 0.85303546  
 0.85339305 0.85510116 0.85605033 0.85789285 0.85846474 0.85856008  
 0.85932229 0.86078832 0.86445855 0.86461199 0.86702777 0.86720073  
 0.86831493 0.86831493 0.86831493 0.86831493 0.86831493 0.86831493  
 0.86831493 0.86831493 0.86831493 0.86831493 0.86831493 0.86831493  
 0.86831493 0.86831493 0.86831493 0.86831493 0.86831493 0.86831493  
 0.86831493 0.86831493 0.86831493 0.86831493 0.86831493 0.86831493  
 0.86831493 0.86831493 0.86831493 0.86831493 0.87205015 0.87224239  
 0.8745074 0.87611176 0.87623128 0.87789984 0.87916593 0.88165201  
 0.88219968 0.8824558 0.88260967 0.88412917 0.88477687 0.88688744  
 0.88694516 0.88844658 0.89015634 0.89308175 0.89337233 0.89379864  
 0.89385899 0.89720588 0.89988703 0.90014192 0.90309141 0.90353476  
 0.90768366 0.91125551 0.91310452 0.91381208 0.91710252 0.91891503  
 0.92250464 0.92258832 0.92262942 0.92268476 0.92301363 0.92327331  
 0.92496696 0.92564693 0.92576821 0.92581558 0.92801883 0.93041502  
 0.9304477 0.93078968 0.9337506 0.93423658 0.93462065 0.93593077  
 0.93599083 0.93945414 0.94411349 0.94442843 0.9452307 0.9465014  
 0.94761981 0.9481983 0.95013843 0.95143081 0.95183159 0.95248663  
 0.95254923 0.95280926 0.95287492 0.95779448 0.95962256 0.96195078  
 0.96211273 0.96847535 0.96859138 0.96933225 0.9708548 0.97128299  
 0.97164951 0.97312014 0.98112538 0.98658958 0.98951332 0.99058334  
 0.99060632 0.99134859 0.99309459 0.99324075 0.99432567 0.99516378  
 0.99578974 0.99685566 0.99853176 0.99922603 0.99926665 1.00104108  
 1.00942727 1.00996375 1.01024882 1.01651818 1.01667165 1.01990573  
 1.0202296 1.02264178 1.02435303 1.02515017 1.02556075 1.02879621  
 1.03486524 1.03867225 1.03999278 1.0432582 1.04671115 1.04684605  
 1.04713658 1.04944352 1.05146695 1.05217051 1.05283803 1.05296428  
 1.05928798 1.06029344 1.0647975 1.06494853 1.06765746 1.08103179  
 1.08873335 1.09172645 1.09255272 1.09651914 1.09687248 1.09997561  
 1.1028141 1.10470365 1.10655517 1.11272014 1.11358478 1.1144143  
 1.12071966 1.12347073 1.12364156 1.13071307 1.13546882 1.1381582  
 1.14249444 1.14975362 1.15630415 1.15653026 1.16264413 1.16264822  
 1.17350676 1.17993389 1.21970888 1.22081378 1.22955633 1.23540239  
 1.23755635 1.23769541 1.25300525 1.25853013 1.25934733 1.27318048  
 1.27940869 1.28319076 1.29834072 1.336572 1.34163593 1.34533429  
 1.34735085 1.34875708 1.35699918 1.36867131 1.37166418 1.38329291  
 1.39834235 1.4121273 1.42011507 1.42015438 1.4488175 1.534281  
 1.54565714 1.64949736]  
  
Parameter 6: [0.75269434 0.76414274 0.76743227 0.76859693 0.76984358 0.77426445  
 0.77639553 0.77920538 0.77937535 0.77955919 0.78023677 0.78252518  
 0.7861445 0.79368119 0.79569849 0.79665885 0.7967054 0.7969133  
 0.79759465 0.79814119 0.79941405 0.79955351 0.80162839 0.80192637  
 0.80233406 0.80547377 0.80579915 0.80602871 0.80686304 0.8071919  
 0.80808791 0.80822901 0.80852815 0.80853878 0.80887508 0.80985161  
 0.81078496 0.81122869 0.81124742 0.81201873 0.81280454 0.81399666  
 0.81425205 0.81436198 0.81451942 0.81493032 0.81519913 0.81540678  
 0.81550254 0.81554238 0.81591059 0.8160116 0.81630853 0.81639283  
 0.81782297 0.81829618 0.81933632 0.81946073 0.81963508 0.81969801  
 0.82005105 0.82060031 0.82118608 0.82193437 0.82210702 0.82211465  
 0.82277615 0.82293779 0.82368721 0.82378822 0.82390484 0.82410903  
 0.82455686 0.82473319 0.82479729 0.82532197 0.82562252 0.82570298  
 0.82592524 0.82600106 0.82621432 0.82631882 0.82729236 0.82829957  
 0.82859658 0.82976001 0.82982272 0.82993703 0.83096218 0.83147657  
 0.83163365 0.83176044 0.83248555 0.83317641 0.83346669 0.83358297  
 0.83386772 0.83397253 0.83469103 0.83469196 0.83498431 0.83528851  
 0.83548494 0.8358186 0.83590348 0.83603286 0.83644186 0.83656939  
 0.83669078 0.83746132 0.83792497 0.83804399 0.83812041 0.83815389  
 0.83837351 0.83840819 0.83903095 0.84057625 0.84068635 0.8409402  
 0.84105689 0.8411973 0.84131637 0.84145245 0.84163084 0.84170456  
 0.84173059 0.84201299 0.84234629 0.84454402 0.8447242 0.8447306  
 0.84479677 0.84541235 0.84552953 0.84575449 0.84591042 0.8465889  
 0.84724228 0.84735454 0.84763785 0.84781594 0.84815667 0.84845237  
 0.84884406 0.84906564 0.84915785 0.84917406 0.84968299 0.84984837  
 0.85001059 0.8500633 0.85015482 0.85093402 0.85099164 0.85100652  
 0.85152251 0.85153022 0.85155483 0.85211117 0.85283822 0.85349647  
 0.85371763 0.85398797 0.85409078 0.85509569 0.85575193 0.85581961  
 0.85613722 0.8561985 0.85637021 0.85645679 0.8565477 0.85659415  
 0.85670205 0.85688197 0.85713305 0.85734754 0.85769585 0.85905391  
 0.85934321 0.85991967 0.86065527 0.8607742 0.86086396 0.86154403  
 0.86157028 0.86217836 0.86220742 0.86235022 0.86241967 0.86296593  
 0.86313768 0.8633911 0.86397235 0.86421852 0.8642196 0.86437853  
 0.86485587 0.86505352 0.86536957 0.86550457 0.86555043 0.86638653  
 0.86666993 0.86668187 0.86680613 0.86682493 0.86718866 0.86773702  
 0.86790909 0.86835511 0.86836947 0.86840125 0.86851218 0.86914079  
 0.86921863 0.8697149 0.86989143 0.86989865 0.87014651 0.87041632  
 0.87051246 0.87097165 0.87122681 0.87172214 0.87187588 0.87190151  
 0.87210641 0.87250053 0.87348719 0.87352844 0.87417788 0.87422437  
 0.8745239 0.8745239 0.8745239 0.8745239 0.8745239 0.8745239  
 0.8745239 0.8745239 0.8745239 0.8745239 0.8745239 0.8745239  
 0.8745239 0.8745239 0.8745239 0.8745239 0.8745239 0.8745239  
 0.8745239 0.8745239 0.8745239 0.8745239 0.8745239 0.8745239  
 0.8745239 0.8745239 0.8745239 0.8745239 0.87552804 0.87557972  
 0.87618863 0.87661993 0.87665205 0.87710062 0.87744098 0.87810931  
 0.87825654 0.87832539 0.87836676 0.87877524 0.87894937 0.87951675  
 0.87953227 0.87993589 0.88039553 0.88118197 0.88126008 0.88137469  
 0.88139091 0.88229066 0.88301143 0.88307995 0.88387286 0.88399205  
 0.8851074 0.88606761 0.88656468 0.8867549 0.88763947 0.88812672  
 0.88909172 0.88911421 0.88912526 0.88914014 0.88922855 0.88929836  
 0.88975366 0.88993646 0.88996906 0.8899818 0.8905741 0.89121826  
 0.89122705 0.89131898 0.89211497 0.89224561 0.89234886 0.89270106  
 0.89271721 0.89364825 0.89490082 0.89498549 0.89520116 0.89554276  
 0.89584343 0.89599894 0.89652051 0.89686794 0.89697568 0.89715177  
 0.8971686 0.8972385 0.89725616 0.89857868 0.89907012 0.89969602  
 0.89973956 0.90145002 0.90148121 0.90168038 0.90208969 0.9022048  
 0.90230333 0.90269868 0.90485072 0.90631967 0.90710565 0.90739331  
 0.90739948 0.90759903 0.90806841 0.9081077 0.90839936 0.90862467  
 0.90879294 0.9090795 0.90953008 0.90971672 0.90972764 0.91020466  
 0.91245912 0.91260334 0.91267998 0.91436537 0.91440662 0.91527604  
 0.91536311 0.91601157 0.91647161 0.9166859 0.91679628 0.91766607  
 0.9192976 0.92032104 0.92067604 0.92155388 0.92248214 0.9225184  
 0.92259651 0.92321668 0.92376064 0.92394978 0.92412923 0.92416317  
 0.92586317 0.92613346 0.92734429 0.92738489 0.92811313 0.93170855  
 0.93377896 0.9345836 0.93480572 0.93587202 0.935967 0.93680122  
 0.93756429 0.93807226 0.93857 0.94022733 0.94045977 0.94068277  
 0.94237784 0.94311741 0.94316333 0.94506437 0.94634285 0.94706584  
 0.94823155 0.95018304 0.95194401 0.9520048 0.95364839 0.95364949  
 0.95656859 0.95829639 0.9689891 0.96928613 0.97163639 0.97320798  
 0.97378703 0.97382442 0.97794016 0.97942541 0.9796451 0.98336386  
 0.98503819 0.98605493 0.99012769 1.0004054 1.00176674 1.00276097  
 1.00330308 1.00368111 1.00589683 1.00903465 1.00983923 1.01296538  
 1.01701111 1.02071692 1.02286427 1.02287484 1.03058034 1.05355549  
 1.05661373 1.08452909]

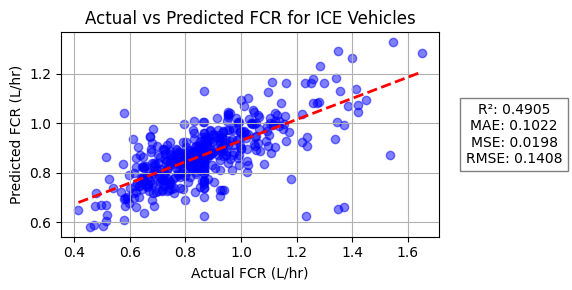
"""  
Scenario 2 for ICE  
Performed linear regression analysis using selected features to predict FCR for ICE vehicles.  
Plotted Actual vs Predicted FCR values and regression line.  
Displayed evaluation metrics: R², MAE, MSE, and RMSE on the plot.  
"""  
  
features = ['Latitude[deg]','Longitude[deg]','Vehicle Speed[km/h]','Distance[km]', 'Engine RPM[RPM]', 'OAT[DegC]', 'Generalized\_Weight']  
target = 'FCR'  
X = df\_ICE[features]  
y = df\_ICE[target]  
model, X\_test, y\_test, y\_pred, regression\_line\_model = linear\_regression\_analysis(features, target, X, y)  
  
# plotting the results  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error  
  
y\_test\_sorted = np.sort(y\_test)  
y\_line = regression\_line\_model.predict(y\_test\_sorted.reshape(-1, 1))  
  
plt.figure(figsize=(6, 3))  
plt.scatter(y\_test, y\_pred, alpha=0.5, color='blue')  
# Plotted the regression line  
plt.plot(y\_test\_sorted, y\_line, color='red', linestyle='--', linewidth=2, label='Regression Line')  
plt.title('Actual vs Predicted FCR for ICE Vehicles')  
plt.xlabel('Actual FCR (L/hr)')  
plt.ylabel('Predicted FCR (L/hr)')  
plt.text(1.20, 0.5, f'R²: {r2\_score(y\_test, y\_pred):.4f}\nMAE: {mean\_absolute\_error(y\_test, y\_pred):.4f}\nMSE: {mean\_squared\_error(y\_test, y\_pred):.4f}\nRMSE: {np.sqrt(mean\_squared\_error(y\_test, y\_pred)):.4f}',  
 fontsize=10, ha='center', va='center', transform=plt.gca().transAxes, bbox=dict(facecolor='white', alpha=0.5))  
plt.grid(True)  
plt.tight\_layout()  
plt.show()

Model Coefficients:  
 Latitude[deg]: 0.3966  
 Longitude[deg]: 0.3935  
 Vehicle Speed[km/h]: 0.0076  
 Distance[km]: -0.0022  
 Engine RPM[RPM]: 0.0002  
 OAT[DegC]: 0.0013  
 Generalized\_Weight: 0.0002  
  
Regression Equation:  
 Slope of the regression line: [ 3.96637813e-01 3.93531578e-01 7.62270412e-03 -2.19456742e-03  
 1.60310069e-04 1.34480393e-03 1.78382168e-04]  
 Intercept: 15.9109  
 Target Variable: FCR  
 FCR = 0.3966\*Latitude[deg] + 0.3935\*Longitude[deg] + 0.0076\*Vehicle Speed[km/h] + -0.0022\*Distance[km] + 0.0002\*Engine RPM[RPM] + 0.0013\*OAT[DegC] + 0.0002\*Generalized\_Weight + 15.9109  
  
Evaluation Metrics:  
 R² Score : 0.4843  
 MAE : 0.1028  
 MSE : 0.0201  
 RMSE : 0.1417



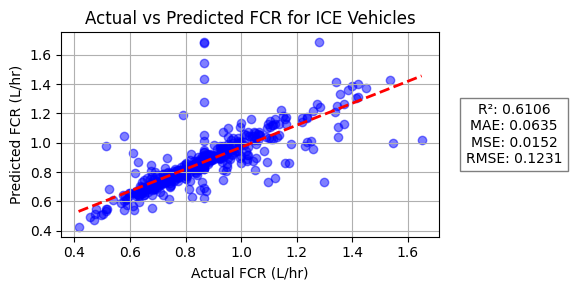
"""  
Scenario 3 for ICE  
Performed linear regression analysis using selected features to predict FCR for ICE vehicles.  
Plotted Actual vs Predicted FCR values and regression line.  
Displayed evaluation metrics: R², MAE, MSE, and RMSE on the plot.  
"""  
  
features = ['Vehicle Speed[km/h]', 'Distance[km]', 'Engine RPM[RPM]', 'OAT[DegC]', 'Generalized\_Weight']  
#features = ['Vehicle Speed[km/h]', 'Distance[km]', 'Engine RPM[RPM]', 'OAT[DegC]']  
target = 'FCR'  
X = df\_ICE[features]  
y = df\_ICE[target]  
model, X\_test, y\_test, y\_pred, regression\_line\_model = linear\_regression\_analysis(features, target, X, y)  
  
# plotting the results  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error  
  
y\_test\_sorted = np.sort(y\_test)  
y\_line = regression\_line\_model.predict(y\_test\_sorted.reshape(-1, 1))  
  
plt.figure(figsize=(6, 3))  
plt.scatter(y\_test, y\_pred, alpha=0.5, color='blue')  
# Plotted the regression line  
plt.plot(y\_test\_sorted, y\_line, color='red', linestyle='--', linewidth=2, label='Regression Line')  
plt.title('Actual vs Predicted FCR for ICE Vehicles')  
plt.xlabel('Actual FCR (L/hr)')  
plt.ylabel('Predicted FCR (L/hr)')  
plt.text(1.20, 0.5, f'R²: {r2\_score(y\_test, y\_pred):.4f}\nMAE: {mean\_absolute\_error(y\_test, y\_pred):.4f}\nMSE: {mean\_squared\_error(y\_test, y\_pred):.4f}\nRMSE: {np.sqrt(mean\_squared\_error(y\_test, y\_pred)):.4f}',  
 fontsize=10, ha='center', va='center', transform=plt.gca().transAxes, bbox=dict(facecolor='white', alpha=0.5))  
plt.grid(True)  
plt.tight\_layout()  
plt.show()

Model Coefficients:  
 Vehicle Speed[km/h]: 0.0077  
 Distance[km]: -0.0021  
 Engine RPM[RPM]: 0.0002  
 OAT[DegC]: 0.0013  
 Generalized\_Weight: 0.0002  
  
Regression Equation:  
 Slope of the regression line: [ 0.00767829 -0.00211231 0.00015705 0.00125734 0.00017501]  
 Intercept: -0.2587  
 Target Variable: FCR  
 FCR = 0.0077\*Vehicle Speed[km/h] + -0.0021\*Distance[km] + 0.0002\*Engine RPM[RPM] + 0.0013\*OAT[DegC] + 0.0002\*Generalized\_Weight + -0.2587  
  
Evaluation Metrics:  
 R² Score : 0.4905  
 MAE : 0.1022  
 MSE : 0.0198  
 RMSE : 0.1408



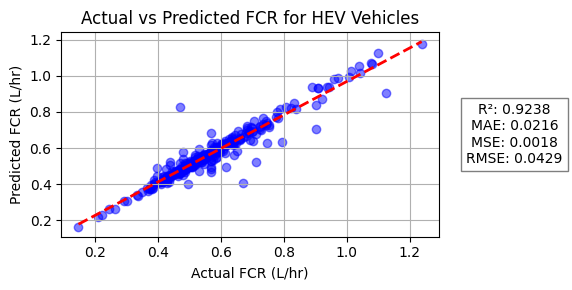
"""  
Scenario 4 for ICE  
Performed linear regression analysis using selected features to predict FCR for ICE vehicles.  
Used features: Vehicle Speed[km/h], Distance[km], Generalized\_Weight, MAF[g/sec], Absolute Load[%], Short Term Fuel Trim Bank 1[%], Short Term Fuel Trim Bank 2[%], Long Term Fuel Trim Bank 1[%], Long Term Fuel Trim Bank 2[%].  
Plotted Actual vs Predicted FCR values and regression line.  
Displayed evaluation metrics: R², MAE, MSE, and RMSE on the plot.  
"""  
  
#features = ['Vehicle Speed[km/h]', 'Distance[km]', 'Engine RPM[RPM]', 'OAT[DegC]', 'Generalized\_Weight','MAF[g/sec]',  
# 'Absolute Load[%]', 'Short Term Fuel Trim Bank 1[%]',  
# 'Short Term Fuel Trim Bank 2[%]', 'Long Term Fuel Trim Bank 1[%]',  
# 'Long Term Fuel Trim Bank 2[%]']  
features = ['Vehicle Speed[km/h]', 'Distance[km]', 'Generalized\_Weight','MAF[g/sec]',  
 'Absolute Load[%]', 'Short Term Fuel Trim Bank 1[%]',  
 'Short Term Fuel Trim Bank 2[%]', 'Long Term Fuel Trim Bank 1[%]',  
 'Long Term Fuel Trim Bank 2[%]']  
target = 'FCR'  
X = df\_ICE[features]  
y = df\_ICE[target]  
model, X\_test, y\_test, y\_pred, regression\_line\_model = linear\_regression\_analysis(features, target, X, y)  
  
# plotting the results  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error  
  
y\_test\_sorted = np.sort(y\_test)  
y\_line = regression\_line\_model.predict(y\_test\_sorted.reshape(-1, 1))  
  
plt.figure(figsize=(6, 3))  
plt.scatter(y\_test, y\_pred, alpha=0.5, color='blue')  
# Plotted the regression line  
plt.plot(y\_test\_sorted, y\_line, color='red', linestyle='--', linewidth=2, label='Regression Line')  
plt.title('Actual vs Predicted FCR for ICE Vehicles')  
plt.xlabel('Actual FCR (L/hr)')  
plt.ylabel('Predicted FCR (L/hr)')  
plt.text(1.20, 0.5, f'R²: {r2\_score(y\_test, y\_pred):.4f}\nMAE: {mean\_absolute\_error(y\_test, y\_pred):.4f}\nMSE: {mean\_squared\_error(y\_test, y\_pred):.4f}\nRMSE: {np.sqrt(mean\_squared\_error(y\_test, y\_pred)):.4f}',  
 fontsize=10, ha='center', va='center', transform=plt.gca().transAxes, bbox=dict(facecolor='white', alpha=0.5))  
plt.grid(True)  
plt.tight\_layout()  
plt.show()

Model Coefficients:  
 Vehicle Speed[km/h]: 0.0029  
 Distance[km]: -0.0010  
 Generalized\_Weight: 0.0001  
 MAF[g/sec]: 0.0485  
 Absolute Load[%]: 0.0035  
 Short Term Fuel Trim Bank 1[%]: -0.0022  
 Short Term Fuel Trim Bank 2[%]: 0.0013  
 Long Term Fuel Trim Bank 1[%]: 0.0083  
 Long Term Fuel Trim Bank 2[%]: 0.0022  
  
Regression Equation:  
 Slope of the regression line: [ 2.91205095e-03 -9.71297358e-04 5.07330125e-05 4.85228489e-02  
 3.47002551e-03 -2.23791788e-03 1.25872051e-03 8.26195897e-03  
 2.21058683e-03]  
 Intercept: -0.1387  
 Target Variable: FCR  
 FCR = 0.0029\*Vehicle Speed[km/h] + -0.0010\*Distance[km] + 0.0001\*Generalized\_Weight + 0.0485\*MAF[g/sec] + 0.0035\*Absolute Load[%] + -0.0022\*Short Term Fuel Trim Bank 1[%] + 0.0013\*Short Term Fuel Trim Bank 2[%] + 0.0083\*Long Term Fuel Trim Bank 1[%] + 0.0022\*Long Term Fuel Trim Bank 2[%] + -0.1387  
  
Evaluation Metrics:  
 R² Score : 0.6106  
 MAE : 0.0635  
 MSE : 0.0152  
 RMSE : 0.1231



"""  
Scenario 1 for HEV  
Performed linear regression analysis for HEV vehicles using 'MAF[g/sec]' as the feature and 'FCR' as the target.  
Trained the model and generated predictions on the test set.  
Plotted Actual vs Predicted FCR values and the regression line.  
Displayed evaluation metrics: R², MAE, MSE, and RMSE on the plot.  
"""  
  
'''  
features = ['Vehicle Speed[km/h]',  
 'Distance[km]',  
 'Engine RPM[RPM]',  
 'OAT[DegC]',  
 'Generalized\_Weight',  
 'HV Battery Power[Watts]',  
 'MAF[g/sec]',  
 'Absolute Load[%]', 'Short Term Fuel Trim Bank 1[%]',  
 'Short Term Fuel Trim Bank 2[%]', 'Long Term Fuel Trim Bank 1[%]',  
 'Long Term Fuel Trim Bank 2[%]'  
 ]  
'''  
'''  
features = ['Vehicle Speed[km/h]',  
 'Distance[km]',  
 'Generalized\_Weight',  
 'MAF[g/sec]',  
 'Absolute Load[%]', 'Short Term Fuel Trim Bank 1[%]',  
 'Long Term Fuel Trim Bank 1[%]'  
 ]  
'''  
  
  
  
features = [  
 'MAF[g/sec]'  
]  
  
target = 'FCR'  
X = df\_HEV[features]  
y = df\_HEV[target]  
model, X\_test, y\_test, y\_pred, regression\_line\_model = linear\_regression\_analysis(features, target, X, y)  
  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error  
  
y\_test\_sorted = np.sort(y\_test)  
y\_line = regression\_line\_model.predict(y\_test\_sorted.reshape(-1, 1))  
  
plt.figure(figsize=(6, 3))  
plt.scatter(y\_test, y\_pred, alpha=0.5, color='blue')  
plt.plot(y\_test\_sorted, y\_line, color='red', linestyle='--', linewidth=2, label='Regression Line')  
plt.title('Actual vs Predicted FCR for HEV Vehicles')  
plt.xlabel('Actual FCR (L/hr)')  
plt.ylabel('Predicted FCR (L/hr)')  
plt.text(1.20, 0.5, f'R²: {r2\_score(y\_test, y\_pred):.4f}\nMAE: {mean\_absolute\_error(y\_test, y\_pred):.4f}\nMSE: {mean\_squared\_error(y\_test, y\_pred):.4f}\nRMSE: {np.sqrt(mean\_squared\_error(y\_test, y\_pred)):.4f}',  
 fontsize=10, ha='center', va='center', transform=plt.gca().transAxes, bbox=dict(facecolor='white', alpha=0.5))  
plt.grid(True)  
plt.tight\_layout()  
plt.show()

Model Coefficients:  
 MAF[g/sec]: 0.0664  
  
Regression Equation:  
 Slope of the regression line: [0.06642316]  
 Intercept: 0.0140  
 Target Variable: FCR  
 FCR = 0.0664\*MAF[g/sec] + 0.0140  
  
Evaluation Metrics:  
 R² Score : 0.9238  
 MAE : 0.0216  
 MSE : 0.0018  
 RMSE : 0.0429

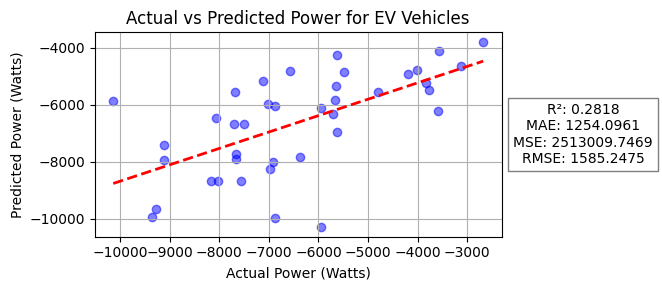


'''  
Created a function to handle outliers using IQR  
'''  
  
def handle\_outliers\_iqr(df, columns, method='cap'):  
 """  
 Detects and handles outliers in specified columns using the IQR method.  
 Adds boolean columns for outlier flags and modifies outliers in place.  
 method: 'cap' (default) replaces outliers with nearest bound, 'remove' drops outlier rows.  
 Returns the modified DataFrame.  
 """  
 df = df.copy()  
 for col in columns:  
 Q1 = df[col].quantile(0.25)  
 Q3 = df[col].quantile(0.75)  
 IQR = Q3 - Q1  
 lower = Q1 - 1.5 \* IQR  
 upper = Q3 + 1.5 \* IQR  
 outlier\_flag = (df[col] < lower) | (df[col] > upper)  
 df[f'{col}\_outlier'] = outlier\_flag  
 if method == 'cap':  
 df.loc[df[col] < lower, col] = lower  
 df.loc[df[col] > upper, col] = upper  
 elif method == 'remove':  
 df = df[~outlier\_flag]  
 return df  
  
# Example usage for EV outliers (capping outliers)  
outlier\_columns = ['Air Conditioning Power[Watts]','Heater Power[Watts]','Vehicle Speed[km/h]','HV Battery Power[Watts]']  
df\_EV\_clean = handle\_outliers\_iqr(df\_EV, outlier\_columns, method='cap')  
df\_EV\_clean.head()

Trip Vehicle Type Latitude[deg] Longitude[deg] \  
1072 554 EV 42.244061 -83.736726   
1097 565 EV 42.243105 -83.721433   
2780 1175 EV 42.254807 -83.725073   
2689 1143 EV 42.244349 -83.732306   
1509 719 EV 42.244535 -83.718833   
  
 Air Conditioning Power[Watts] Heater Power[Watts] Vehicle Speed[km/h] \  
1072 111.184211 187.500000 48.529999   
1097 0.000000 0.000000 53.631999   
2780 471.176471 528.431373 8.471024   
2689 0.000000 0.000000 56.034837   
1509 260.483871 152.419355 27.739903   
  
 Distance[km] Engine RPM[RPM] OAT[DegC] ... MAF[g/sec] \  
1072 0.927557 0.0 11.500000 ... 0.0   
1097 0.519562 0.0 10.000000 ... 0.0   
2780 0.184437 0.0 2.817647 ... 0.0   
2689 0.544876 0.0 15.000000 ... 0.0   
1509 1.526925 0.0 8.159677 ... 0.0   
  
 Absolute Load[%] Short Term Fuel Trim Bank 1[%] \  
1072 0.0 0.0   
1097 0.0 0.0   
2780 0.0 0.0   
2689 0.0 0.0   
1509 0.0 0.0   
  
 Short Term Fuel Trim Bank 2[%] Long Term Fuel Trim Bank 1[%] \  
1072 0.0 0.0   
1097 0.0 0.0   
2780 0.0 0.0   
2689 0.0 0.0   
1509 0.0 0.0   
  
 Long Term Fuel Trim Bank 2[%] Air Conditioning Power[Watts]\_outlier \  
1072 0.0 False   
1097 0.0 False   
2780 0.0 False   
2689 0.0 False   
1509 0.0 False   
  
 Heater Power[Watts]\_outlier Vehicle Speed[km/h]\_outlier \  
1072 False False   
1097 False False   
2780 False True   
2689 False False   
1509 False False   
  
 HV Battery Power[Watts]\_outlier   
1072 True   
1097 False   
2780 False   
2689 False   
1509 False   
  
[5 rows x 23 columns]

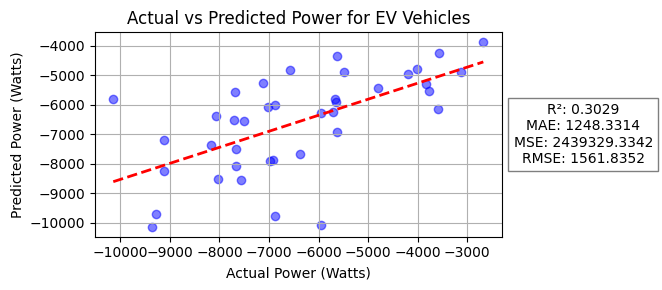
'''  
Scenario 1 for EV  
Selected features for linear regression analysis to predict HV Battery Power for EVs.  
Performed linear regression using the selected features and target.  
Plotted Actual vs Predicted Power for EV vehicles, including regression line and evaluation metrics (R², MAE, MSE, RMSE).  
'''  
  
'''  
features = ['Latitude[deg]',  
 'Longitude[deg]',  
 'Air Conditioning Power[Watts]',  
 'Heater Power[Watts]',  
 'Vehicle Speed[km/h]',  
 'Distance[km]',  
 'OAT[DegC]',  
 'Generalized\_Weight']  
'''  
  
'''  
features = [  
 'Air Conditioning Power[Watts]',  
 'Heater Power[Watts]',  
 'Vehicle Speed[km/h]',  
 'Distance[km]',  
 'OAT[DegC]']  
'''  
  
features = [  
 'Air Conditioning Power[Watts]',  
 'Heater Power[Watts]',  
 'Vehicle Speed[km/h]',  
]  
  
target = 'HV Battery Power[Watts]'  
X = df\_EV[features]  
y = df\_EV[target]  
model, X\_test, y\_test, y\_pred, regression\_line\_model = linear\_regression\_analysis(features, target, X, y)  
  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error  
  
y\_test\_sorted = np.sort(y\_test)  
y\_line = regression\_line\_model.predict(y\_test\_sorted.reshape(-1, 1))  
  
plt.figure(figsize=(7, 3))  
plt.scatter(y\_test, y\_pred, alpha=0.5, color='blue')  
plt.plot(y\_test\_sorted, y\_line, color='red', linestyle='--', linewidth=2, label='Regression Line')  
plt.title('Actual vs Predicted Power for EV Vehicles')  
plt.xlabel('Actual Power (Watts)')  
plt.ylabel('Predicted Power (Watts)')  
plt.text(  
 1.20, 0.5,  
 f'R²: {r2\_score(y\_test, y\_pred):.4f}\nMAE: {mean\_absolute\_error(y\_test, y\_pred):.4f}\nMSE: {mean\_squared\_error(y\_test, y\_pred):.4f}\nRMSE: {np.sqrt(mean\_squared\_error(y\_test, y\_pred)):.4f}',  
 fontsize=10, ha='center', va='center', transform=plt.gca().transAxes,  
 bbox=dict(facecolor='white', alpha=0.5)  
)  
plt.grid(True)  
plt.tight\_layout()  
plt.show()

Model Coefficients:  
 Air Conditioning Power[Watts]: -1.8823  
 Heater Power[Watts]: -1.4577  
 Vehicle Speed[km/h]: -105.0709  
  
Regression Equation:  
 Slope of the regression line: [ -1.88227038 -1.45771702 -105.07093857]  
 Intercept: -643.8252  
 Target Variable: HV Battery Power[Watts]  
 HV Battery Power[Watts] = -1.8823\*Air Conditioning Power[Watts] + -1.4577\*Heater Power[Watts] + -105.0709\*Vehicle Speed[km/h] + -643.8252  
  
Evaluation Metrics:  
 R² Score : 0.2818  
 MAE : 1254.0961  
 MSE : 2513009.7469  
 RMSE : 1585.2475



'''  
Scenario 2 for EV without outliers  
'''  
  
'''  
features = ['Latitude[deg]',  
 'Longitude[deg]',  
 'Air Conditioning Power[Watts]',  
 'Heater Power[Watts]',  
 'Vehicle Speed[km/h]',  
 'Distance[km]',  
 'OAT[DegC]',  
 'Generalized\_Weight']  
'''  
  
'''  
features = [  
 'Air Conditioning Power[Watts]',  
 'Heater Power[Watts]',  
 'Vehicle Speed[km/h]',  
 'Distance[km]',  
 'OAT[DegC]']  
'''  
  
  
features = [  
 'Air Conditioning Power[Watts]',  
 'Heater Power[Watts]',  
 'Vehicle Speed[km/h]',  
 ]  
  
  
target = 'HV Battery Power[Watts]'  
X = df\_EV\_clean[features]  
y = df\_EV\_clean[target]  
model, X\_test, y\_test, y\_pred, regression\_line\_model = linear\_regression\_analysis(features, target, X, y)  
  
# plotting the results  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error  
  
y\_test\_sorted = np.sort(y\_test)  
y\_line = regression\_line\_model.predict(y\_test\_sorted.reshape(-1, 1))  
  
plt.figure(figsize=(7, 3))  
plt.scatter(y\_test, y\_pred, alpha=0.5, color='blue')  
#plot the regression line  
plt.plot(y\_test\_sorted, y\_line, color='red', linestyle='--', linewidth=2, label='Regression Line')  
plt.title('Actual vs Predicted Power for EV Vehicles')  
plt.xlabel('Actual Power (Watts)')  
plt.ylabel('Predicted Power (Watts)')  
plt.text(1.20, 0.5, f'R²: {r2\_score(y\_test, y\_pred):.4f}\nMAE: {mean\_absolute\_error(y\_test, y\_pred):.4f}\nMSE: {mean\_squared\_error(y\_test, y\_pred):.4f}\nRMSE: {np.sqrt(mean\_squared\_error(y\_test, y\_pred)):.4f}',  
 fontsize=10, ha='center', va='center', transform=plt.gca().transAxes, bbox=dict(facecolor='white', alpha=0.5))  
plt.grid(True)  
plt.tight\_layout()  
plt.show()

Model Coefficients:  
 Air Conditioning Power[Watts]: -1.5130  
 Heater Power[Watts]: -1.6000  
 Vehicle Speed[km/h]: -99.5030  
  
Regression Equation:  
 Slope of the regression line: [ -1.51298951 -1.60004374 -99.50303112]  
 Intercept: -969.1184  
 Target Variable: HV Battery Power[Watts]  
 HV Battery Power[Watts] = -1.5130\*Air Conditioning Power[Watts] + -1.6000\*Heater Power[Watts] + -99.5030\*Vehicle Speed[km/h] + -969.1184  
  
Evaluation Metrics:  
 R² Score : 0.3029  
 MAE : 1248.3314  
 MSE : 2439329.3342  
 RMSE : 1561.8352



df\_EV.describe()

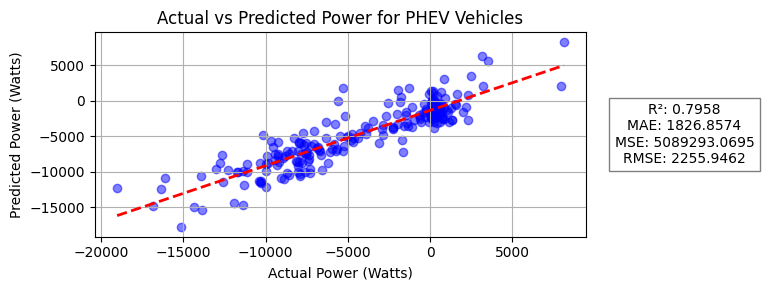
Trip Latitude[deg] Longitude[deg] \  
count 192.000000 192.000000 192.000000   
mean 1312.286458 42.271497 -83.735299   
std 520.314309 0.016914 0.021729   
min 554.000000 42.231426 -83.796345   
25% 822.750000 42.261343 -83.746096   
50% 1177.000000 42.272274 -83.737289   
75% 1805.000000 42.284359 -83.724456   
max 2200.000000 42.308496 -83.679000   
  
 Air Conditioning Power[Watts] Heater Power[Watts] \  
count 192.000000 192.000000   
mean 457.753092 758.677578   
std 387.748865 953.331492   
min 0.000000 0.000000   
25% 75.323201 0.000000   
50% 389.670273 424.481066   
75% 783.016172 1000.000000   
max 1530.806452 3911.167513   
  
 Vehicle Speed[km/h] Distance[km] Engine RPM[RPM] OAT[DegC] \  
count 192.000000 192.000000 192.0 192.000000   
mean 37.601938 2.633718 0.0 1.682723   
std 11.567310 2.456442 0.0 6.019795   
min 3.446784 0.132661 0.0 -14.455868   
25% 30.326173 0.927085 0.0 -2.080354   
50% 36.819866 1.797607 0.0 2.000000   
75% 44.896273 3.430748 0.0 5.145057   
max 67.251182 14.155463 0.0 15.165680   
  
 Generalized\_Weight FCR HV Battery Power[Watts] MAF[g/sec] \  
count 192.0 192.0 192.000000 192.0   
mean 3500.0 0.0 -6523.042672 0.0   
std 0.0 0.0 2324.714219 0.0   
min 3500.0 0.0 -14409.176136 0.0   
25% 3500.0 0.0 -7680.062309 0.0   
50% 3500.0 0.0 -6503.715555 0.0   
75% 3500.0 0.0 -4900.811702 0.0   
max 3500.0 0.0 482.351974 0.0   
  
 Absolute Load[%] Short Term Fuel Trim Bank 1[%] \  
count 192.0 192.0   
mean 0.0 0.0   
std 0.0 0.0   
min 0.0 0.0   
25% 0.0 0.0   
50% 0.0 0.0   
75% 0.0 0.0   
max 0.0 0.0   
  
 Short Term Fuel Trim Bank 2[%] Long Term Fuel Trim Bank 1[%] \  
count 192.0 192.0   
mean 0.0 0.0   
std 0.0 0.0   
min 0.0 0.0   
25% 0.0 0.0   
50% 0.0 0.0   
75% 0.0 0.0   
max 0.0 0.0   
  
 Long Term Fuel Trim Bank 2[%]   
count 192.0   
mean 0.0   
std 0.0   
min 0.0   
25% 0.0   
50% 0.0   
75% 0.0   
max 0.0

df\_PHEV.describe()

Trip Latitude[deg] Longitude[deg] \  
count 1072.000000 1072.000000 1072.000000   
mean 1068.349813 42.273535 -83.724487   
std 595.376940 0.018781 0.024099   
min 2.000000 42.229924 -83.799773   
25% 591.750000 42.259781 -83.739367   
50% 1027.500000 42.277576 -83.721872   
75% 1505.250000 42.285169 -83.708786   
max 2497.000000 42.318748 -83.677901   
  
 Air Conditioning Power[Watts] Heater Power[Watts] \  
count 1072.000000 1072.0   
mean 23.886959 0.0   
std 115.817200 0.0   
min 0.000000 0.0   
25% 0.000000 0.0   
50% 0.000000 0.0   
75% 0.000000 0.0   
max 1854.489437 0.0   
  
 Vehicle Speed[km/h] Distance[km] Engine RPM[RPM] OAT[DegC] \  
count 1072.000000 1072.000000 1072.000000 1072.000000   
mean 42.361808 3.771616 480.276407 1.194308   
std 11.850411 3.752166 460.683247 6.459601   
min 11.558719 0.037676 0.000000 -20.871176   
25% 35.242796 1.734695 0.000000 -2.776627   
50% 41.526763 3.064417 465.831741 2.016203   
75% 47.297805 4.449652 791.785924 5.473130   
max 101.218062 60.774807 3360.000000 26.222477   
  
 Generalized\_Weight FCR HV Battery Power[Watts] MAF[g/sec] \  
count 1072.000000 1072.000000 1072.000000 1072.000000   
mean 3778.348617 1.300838 -4714.758687 4.607863   
std 382.654582 1.184929 5009.957336 3.296976   
min 3000.000000 0.000000 -23445.889629 0.170000   
25% 3635.765692 1.115816 -8076.683138 1.736842   
50% 4000.000000 1.300838 -4519.362083 4.607863   
75% 4000.000000 1.300838 -145.980815 6.140789   
max 4000.000000 8.519646 8125.363238 40.230000   
  
 Absolute Load[%] Short Term Fuel Trim Bank 1[%] \  
count 1072.000000 1072.000000   
mean 14.257791 0.312255   
std 0.616130 0.013494   
min 0.000000 0.000000   
25% 14.257791 0.312255   
50% 14.257791 0.312255   
75% 14.257791 0.312255   
max 28.515582 0.624510   
  
 Short Term Fuel Trim Bank 2[%] Long Term Fuel Trim Bank 1[%] \  
count 1072.0 1072.000000   
mean 0.0 -2.304565   
std 0.0 0.001693   
min 0.0 -2.343750   
25% 0.0 -2.304565   
50% 0.0 -2.304565   
75% 0.0 -2.304565   
max 0.0 -2.265380   
  
 Long Term Fuel Trim Bank 2[%]   
count 1072.0   
mean 0.0   
std 0.0   
min 0.0   
25% 0.0   
50% 0.0   
75% 0.0   
max 0.0

'''  
Scenario 1 for PHEV  
Selected features and target for linear regression analysis on PHEV data.  
Performed linear regression using the selected features to predict HV Battery Power.  
Plotted Actual vs Predicted Power for PHEV vehicles, including regression line and evaluation metrics (R², MAE, MSE, RMSE).  
'''  
  
'''  
features = ['Latitude[deg]',  
 'Longitude[deg]',  
 'Engine RPM[RPM]',  
 'Air Conditioning Power[Watts]',  
 'Heater Power[Watts]',  
 'Vehicle Speed[km/h]',  
 'Distance[km]',  
 'OAT[DegC]',  
 'Generalized\_Weight',  
 ]  
'''  
  
features = [  
 'Engine RPM[RPM]',  
 'Air Conditioning Power[Watts]',  
 'Vehicle Speed[km/h]',  
 'OAT[DegC]',  
]  
  
target = 'HV Battery Power[Watts]'  
X = df\_PHEV[features]  
y = df\_PHEV[target]  
model, X\_test, y\_test, y\_pred, regression\_line\_model = linear\_regression\_analysis(features, target, X, y)  
  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.metrics import r2\_score, mean\_absolute\_error, mean\_squared\_error  
  
y\_test\_sorted = np.sort(y\_test)  
y\_line = regression\_line\_model.predict(y\_test\_sorted.reshape(-1, 1))  
  
plt.figure(figsize=(8, 3))  
plt.scatter(y\_test, y\_pred, alpha=0.5, color='blue')  
plt.plot(y\_test\_sorted, y\_line, color='red', linestyle='--', linewidth=2, label='Regression Line')  
plt.title('Actual vs Predicted Power for PHEV Vehicles')  
plt.xlabel('Actual Power (Watts)')  
plt.ylabel('Predicted Power (Watts)')  
plt.text(  
 1.20, 0.5,  
 f'R²: {r2\_score(y\_test, y\_pred):.4f}\nMAE: {mean\_absolute\_error(y\_test, y\_pred):.4f}\nMSE: {mean\_squared\_error(y\_test, y\_pred):.4f}\nRMSE: {np.sqrt(mean\_squared\_error(y\_test, y\_pred)):.4f}',  
 fontsize=10, ha='center', va='center', transform=plt.gca().transAxes,  
 bbox=dict(facecolor='white', alpha=0.5)  
)  
plt.grid(True)  
plt.tight\_layout()  
plt.show()

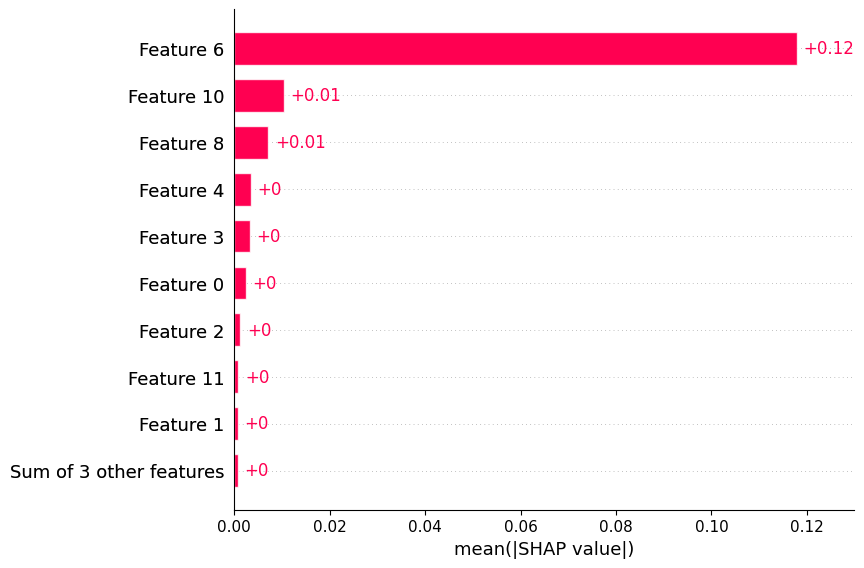
Model Coefficients:  
 Engine RPM[RPM]: 8.8037  
 Air Conditioning Power[Watts]: -0.5881  
 Vehicle Speed[km/h]: -198.5269  
 OAT[DegC]: 144.6615  
  
Regression Equation:  
 Slope of the regression line: [ 8.80374758 -0.58811294 -198.5268572 144.66146801]  
 Intercept: -774.3902  
 Target Variable: HV Battery Power[Watts]  
 HV Battery Power[Watts] = 8.8037\*Engine RPM[RPM] + -0.5881\*Air Conditioning Power[Watts] + -198.5269\*Vehicle Speed[km/h] + 144.6615\*OAT[DegC] + -774.3902  
  
Evaluation Metrics:  
 R² Score : 0.7958  
 MAE : 1826.8574  
 MSE : 5089293.0695  
 RMSE : 2255.9462

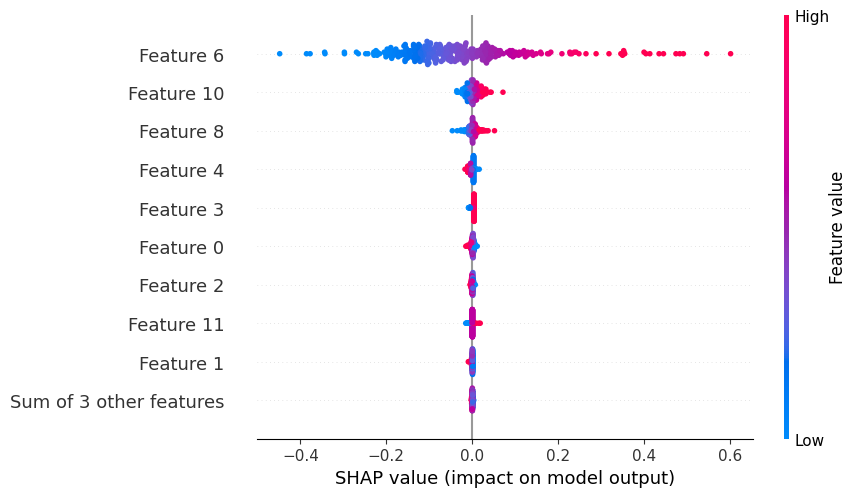


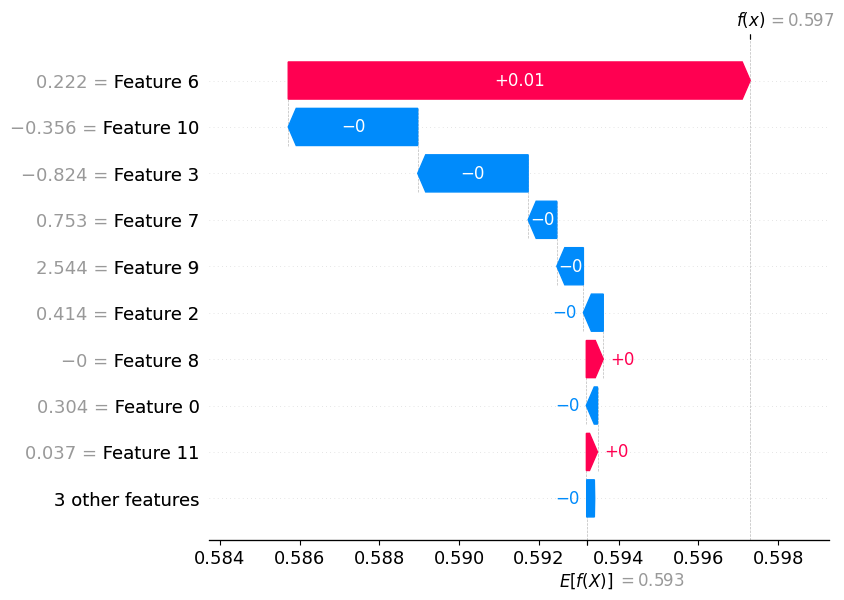
df\_ICE.info()

<class 'pandas.core.frame.DataFrame'>  
Index: 2200 entries, 4973 to 5039  
Data columns (total 19 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Trip 2200 non-null int64   
 1 Vehicle Type 2200 non-null object   
 2 Latitude[deg] 2200 non-null float64  
 3 Longitude[deg] 2200 non-null float64  
 4 Air Conditioning Power[Watts] 2200 non-null float64  
 5 Heater Power[Watts] 0 non-null float64  
 6 Vehicle Speed[km/h] 2200 non-null float64  
 7 Distance[km] 2200 non-null float64  
 8 Engine RPM[RPM] 2200 non-null float64  
 9 OAT[DegC] 2200 non-null float64  
 10 Generalized\_Weight 2200 non-null float64  
 11 FCR 2200 non-null float64  
 12 HV Battery Power[Watts] 2200 non-null float64  
 13 MAF[g/sec] 2200 non-null float64  
 14 Absolute Load[%] 2200 non-null float64  
 15 Short Term Fuel Trim Bank 1[%] 2200 non-null float64  
 16 Short Term Fuel Trim Bank 2[%] 2200 non-null float64  
 17 Long Term Fuel Trim Bank 1[%] 2200 non-null float64  
 18 Long Term Fuel Trim Bank 2[%] 2200 non-null float64  
dtypes: float64(17), int64(1), object(1)  
memory usage: 343.8+ KB

'''  
This cell performs SHAP (SHapley Additive exPlanations) analysis on a linear regression model for HEV vehicles:  
  
1. Imports required libraries for SHAP analysis, linear regression, and data preprocessing  
2. Defines features including vehicle performance metrics, environmental conditions, and fuel system parameters  
3. Sets FCR (Fuel Consumption Rate) as the target variable for HEV vehicles  
4. Standardizes features using StandardScaler to improve SHAP analysis with linear models  
5. Splits data into training and testing sets (80/20 split)  
6. Trains a LinearRegression model on the standardized training data  
7. Creates a SHAP explainer for the linear model using training data  
8. Generates SHAP values for test data to understand feature importance  
9. Visualizes SHAP analysis through:  
 - Bar plot showing overall feature importance  
 - Beeswarm plot showing feature effects distribution  
 - Waterfall plot for individual prediction explanation  
10. Creates a DataFrame of SHAP values for further analysis  
'''  
  
import pandas as pd  
import numpy as np  
import shap  
import matplotlib.pyplot as plt  
from sklearn.linear\_model import LinearRegression  
from sklearn.model\_selection import train\_test\_split  
from sklearn.preprocessing import StandardScaler  
  
# Example features and target  
features = ['Vehicle Speed[km/h]',  
 'Distance[km]',  
 'Engine RPM[RPM]',  
 'OAT[DegC]',  
 'Generalized\_Weight',  
 'HV Battery Power[Watts]',  
 'MAF[g/sec]',  
 'Absolute Load[%]', 'Short Term Fuel Trim Bank 1[%]',  
 'Short Term Fuel Trim Bank 2[%]', 'Long Term Fuel Trim Bank 1[%]',  
 'Long Term Fuel Trim Bank 2[%]'  
 ]  
  
target = 'FCR'  
X = df\_HEV[features]  
y = df\_HEV[target]  
  
# Standardize features (optional but helps SHAP with linear models)  
scaler = StandardScaler()  
X\_scaled = scaler.fit\_transform(X)  
  
# Train-test split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)  
  
model = LinearRegression()  
model.fit(X\_train, y\_train)  
  
explainer = shap.Explainer(model, X\_train) # uses LinearExplainer  
shap\_values = explainer(X\_test)  
  
shap.plots.bar(shap\_values)  
  
shap.plots.beeswarm(shap\_values)  
  
shap.plots.waterfall(shap\_values[0])  
  
shap\_df = pd.DataFrame(shap\_values.values, columns=features)  
shap\_df.head()







Vehicle Speed[km/h] Distance[km] Engine RPM[RPM] OAT[DegC] \  
0 -0.000282 -0.000022 -0.000499 -0.002771   
1 -0.002052 0.000605 -0.003173 -0.003914   
2 0.003829 -0.000657 0.000621 0.004001   
3 -0.002094 -0.000919 0.001043 -0.003460   
4 -0.002272 0.000614 0.000205 -0.002499   
  
 Generalized\_Weight HV Battery Power[Watts] MAF[g/sec] Absolute Load[%] \  
0 0.000014 0.0 0.011589 -0.000718   
1 0.002829 0.0 0.081227 -0.001715   
2 -0.003796 0.0 -0.067177 0.000735   
3 -0.006601 0.0 0.037575 0.000536   
4 -0.005238 0.0 0.227652 -0.000433   
  
 Short Term Fuel Trim Bank 1[%] Short Term Fuel Trim Bank 2[%] \  
0 0.000426 -0.000667   
1 -0.016759 -0.000056   
2 -0.003601 -0.000056   
3 0.002750 -0.000056   
4 0.011011 -0.000701   
  
 Long Term Fuel Trim Bank 1[%] Long Term Fuel Trim Bank 2[%]   
0 -0.003248 0.000280   
1 -0.012696 0.000168   
2 -0.026412 0.000168   
3 -0.004292 0.000168   
4 0.006995 0.002075

# 7 K means Clustering Classification models to categorize vehicle type

df\_combined = pd.concat([df\_ICE, df\_HEV, df\_PHEV, df\_EV], ignore\_index=True)

df\_combined\_k = df\_combined[['Trip','FCR']]

df\_combined\_sf = df\_combined[['Vehicle Speed[km/h]','FCR']]

df\_combined\_sb = df\_combined[['Vehicle Speed[km/h]','HV Battery Power[Watts]']]

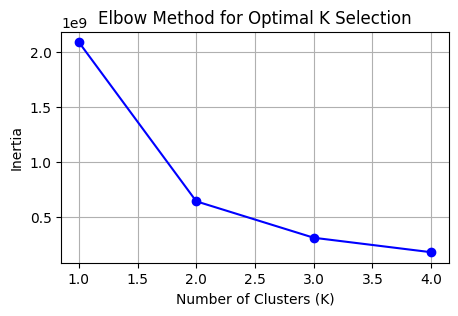
df\_combined\_of = df\_combined[['OAT[DegC]','FCR']]

df\_combined\_ob = df\_combined[['OAT[DegC]','HV Battery Power[Watts]']]

'''  
This cell:  
- Imports necessary libraries: numpy, matplotlib.pyplot, and KMeans from sklearn.  
- Defines a function `plot\_kmeans\_elbow` that:  
 - Computes KMeans clustering inertia for cluster counts from 1 to 4.  
 - Calculates the difference in inertia to suggest an optimal number of clusters (elbow point).  
 - Prints the suggested optimal K.  
 - Plots the inertia values (elbow curve) to visually assist in selecting the optimal number of clusters.  
'''  
  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.cluster import KMeans  
  
def plot\_kmeans\_elbow(df):  
 """  
 Plots the elbow curve for KMeans clustering to help select the optimal number of clusters.  
   
 Args:  
 df (pd.DataFrame): DataFrame containing the features for clustering.  
 """  
 inertia = []  
 for i in range(1, 5):  
 kmeans = KMeans(n\_clusters=i, init='random', random\_state=42)  
 kmeans.fit(df)  
 inertia.append(kmeans.inertia\_)  
  
 # Find elbow point (simple method: where the decrease in inertia slows down the most)  
 diff = np.diff(inertia)  
 elbow\_k = np.argmin(diff) + 2 # +2 because diff is one less and we start from k=1  
 print("Suggested optimal K:", elbow\_k)   
  
 # Plot the elbow curve  
 plt.figure(figsize=(5, 3))  
 plt.plot(range(1, 5), inertia, 'bo-')  
 plt.xlabel('Number of Clusters (K)')  
 plt.ylabel('Inertia')  
 plt.title('Elbow Method for Optimal K Selection')  
 plt.grid(True)  
 plt.show()

plot\_kmeans\_elbow(df\_combined\_k)

Suggested optimal K: 2

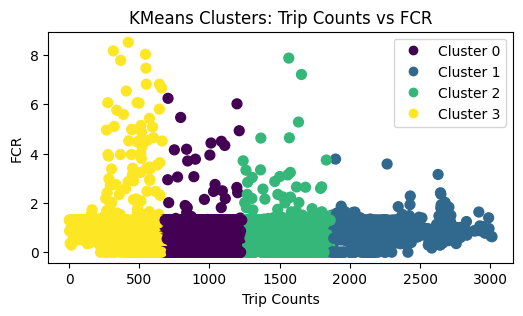


'''  
This cell:  
- Imports KMeans from sklearn.  
- Defines a function `fit\_predict\_kmeans` that:  
 - Fits KMeans clustering to the input DataFrame with a specified number of clusters.  
 - Predicts cluster labels for the input data.  
 - Adds a new 'Cluster' column to the DataFrame with the assigned cluster labels.  
 - Returns the modified DataFrame, the predicted cluster labels, and the fitted KMeans object.  
'''  
  
from sklearn.cluster import KMeans  
  
def fit\_predict\_kmeans(df, n\_clusters, random\_state=42):  
 """  
 Fits KMeans clustering on the given DataFrame and returns a copy with a new 'Cluster' column.  
   
 Parameters:  
 df (pd.DataFrame): The input DataFrame (features only, no target).  
 n\_clusters (int): Number of clusters to use.  
 random\_state (int): Random state for reproducibility.  
   
 Returns:  
 pd.DataFrame: A copy of the input DataFrame with an added 'Cluster' column.  
 np.ndarray: The predicted cluster labels.  
 KMeans: The fitted KMeans object.  
 """  
 kmeans = KMeans(n\_clusters=n\_clusters, random\_state=random\_state)  
 kmeans.fit(df)  
 pred = kmeans.predict(df)  
 df['Cluster'] = kmeans.labels\_  
 return df, pred, kmeans  
  
# Example usage:  
# df\_with\_clusters, cluster\_labels, kmeans\_model = fit\_predict\_kmeans(df, elbow\_k)

fit\_predict\_kmeans(df\_combined\_k, 4)

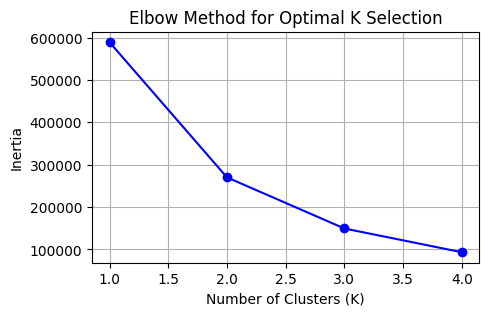
( Trip FCR Cluster  
 0 2628 3.154674 1  
 1 2645 2.411606 1  
 2 2648 2.353453 1  
 3 2678 2.021139 1  
 4 1637 2.015042 2  
 ... ... ... ...  
 5045 1996 0.000000 1  
 5046 728 0.000000 0  
 5047 1674 0.000000 2  
 5048 596 0.000000 3  
 5049 1026 0.000000 0  
   
 [5050 rows x 3 columns],  
 array([1, 1, 1, ..., 2, 3, 0], shape=(5050,), dtype=int32),  
 KMeans(n\_clusters=4, random\_state=42))

'''  
This cell:  
- Creates a scatter plot of 'Trip' vs 'FCR' from df\_combined\_k.  
- Colors the points by their assigned KMeans cluster using the 'Cluster' column.  
- Sets axis labels and a title.  
- Adds a legend for the clusters.  
- Displays the plot.  
'''  
  
plt.figure(figsize=(6, 3))  
scatter = plt.scatter(  
 df\_combined\_k['Trip'],  
 df\_combined\_k['FCR'],  
 c=df\_combined\_k['Cluster'],  
 cmap='viridis',  
 s=50  
)  
plt.xlabel('Trip Counts')  
plt.ylabel('FCR')  
plt.title('KMeans Clusters: Trip Counts vs FCR')  
handles, labels = scatter.legend\_elements(prop="colors")  
plt.legend(handles, [f"Cluster {i}" for i in range(len(handles))])  
plt.show()



plot\_kmeans\_elbow(df\_combined\_sf)

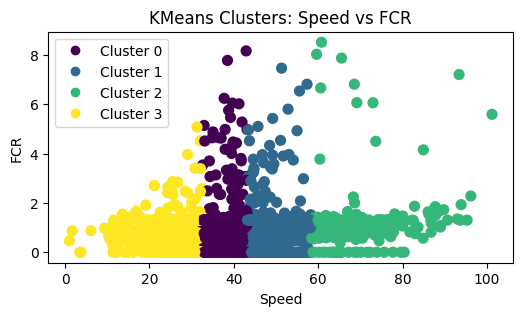
Suggested optimal K: 2



fit\_predict\_kmeans(df\_combined\_sf,4)

( Vehicle Speed[km/h] FCR Cluster  
 0 44.100000 3.154674 1  
 1 24.263889 2.411606 3  
 2 36.552916 2.353453 0  
 3 30.005076 2.021139 3  
 4 45.045161 2.015042 1  
 ... ... ... ...  
 5045 44.174461 0.000000 1  
 5046 64.811233 0.000000 2  
 5047 38.361678 0.000000 0  
 5048 63.285435 0.000000 2  
 5049 58.763749 0.000000 2  
   
 [5050 rows x 3 columns],  
 array([1, 3, 0, ..., 0, 2, 2], shape=(5050,), dtype=int32),  
 KMeans(n\_clusters=4, random\_state=42))

'''  
This cell:  
- Creates a scatter plot of 'Vehicle Speed[km/h]' vs 'FCR' from df\_combined\_sf.  
- Colors the points by their assigned KMeans cluster using the 'Cluster' column.  
- Sets axis labels and a title.  
- Adds a legend for the clusters.  
- Displays the plot.  
'''  
  
plt.figure(figsize=(6, 3))  
scatter = plt.scatter(  
 df\_combined\_sf['Vehicle Speed[km/h]'],  
 df\_combined\_sf['FCR'],  
 c=df\_combined\_sf['Cluster'],  
 cmap='viridis',  
 s=50  
)  
plt.xlabel('Speed')  
plt.ylabel('FCR')  
plt.title('KMeans Clusters: Speed vs FCR')  
handles, labels = scatter.legend\_elements(prop="colors")  
plt.legend(handles, [f"Cluster {i}" for i in range(len(handles))])  
plt.show()



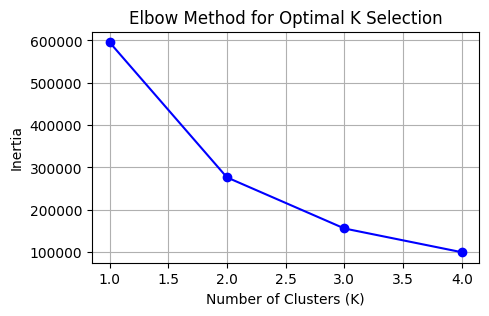
df\_combined\_sf = df\_combined[['Vehicle Type','Vehicle Speed[km/h]','FCR']]  
df\_combined\_sf.head()

Vehicle Type Vehicle Speed[km/h] FCR  
0 ICE 44.100000 3.154674  
1 ICE 24.263889 2.411606  
2 ICE 36.552916 2.353453  
3 ICE 30.005076 2.021139  
4 ICE 45.045161 2.015042

'''  
This cell:  
- Maps the 'Vehicle Type' column in df\_combined\_sf from string labels ('ICE', 'HEV', 'EV', 'PHEV')  
 to numeric codes (0, 1, 2, 3) for further analysis or modeling.  
'''  
  
df\_combined\_sf['Vehicle Type'] = df\_combined\_sf['Vehicle Type'].map({'ICE': 0, 'HEV': 1, 'EV': 2, 'PHEV': 3})

plot\_kmeans\_elbow(df\_combined\_sf)

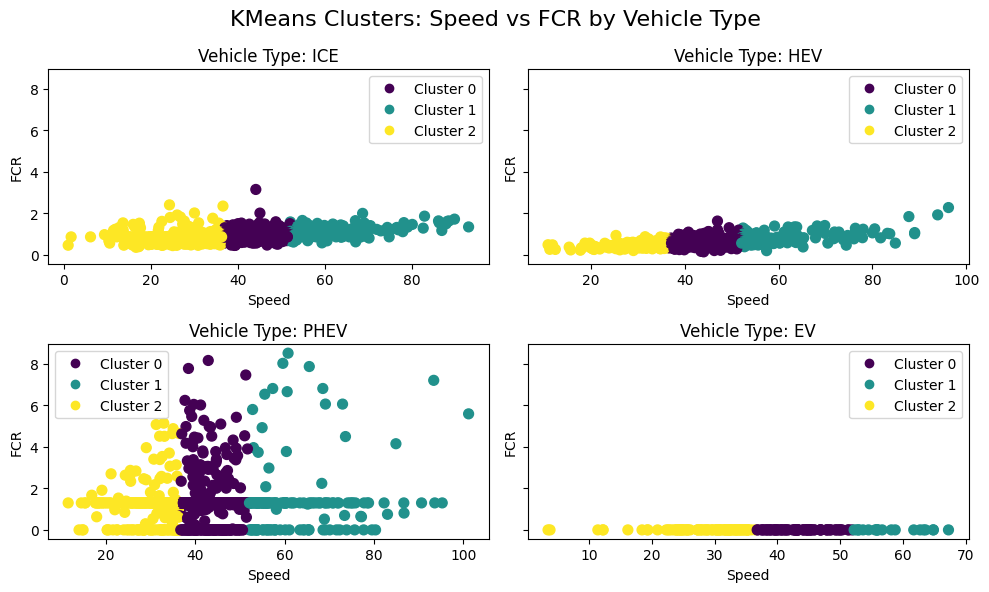
Suggested optimal K: 2



fit\_predict\_kmeans(df\_combined\_sf,3)

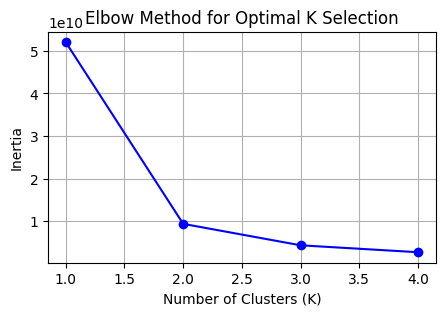
( Vehicle Type Vehicle Speed[km/h] FCR Cluster  
 0 0 44.100000 3.154674 0  
 1 0 24.263889 2.411606 2  
 2 0 36.552916 2.353453 2  
 3 0 30.005076 2.021139 2  
 4 0 45.045161 2.015042 0  
 ... ... ... ... ...  
 5045 2 44.174461 0.000000 0  
 5046 2 64.811233 0.000000 1  
 5047 2 38.361678 0.000000 0  
 5048 2 63.285435 0.000000 1  
 5049 2 58.763749 0.000000 1  
   
 [5050 rows x 4 columns],  
 array([0, 2, 2, ..., 0, 1, 1], shape=(5050,), dtype=int32),  
 KMeans(n\_clusters=3, random\_state=42))

'''  
This code:  
- Imports matplotlib for plotting.  
- Gets the unique vehicle types from the dataframe.  
- Creates a 2x2 grid of subplots for visualizing each vehicle type.  
- Iterates over each vehicle type, filtering the dataframe for that type.  
- Plots a scatter plot of Speed vs FCR for each vehicle type, colored by cluster.  
- Maps numeric vehicle type codes back to string labels for subplot titles.  
- Sets axis labels and adds a legend for clusters in each subplot.  
- Sets a common title for the figure and adjusts layout.  
- Displays the plot.  
'''  
  
import matplotlib.pyplot as plt  
  
vehicle\_types = df\_combined\_sf['Vehicle Type'].unique()  
  
fig, axes = plt.subplots(2, 2, figsize=(10, 6), sharey=True)  
axes = axes.flatten()  
  
for i, vtype in enumerate(vehicle\_types):  
 ax = axes[i]  
 subset = df\_combined\_sf[df\_combined\_sf['Vehicle Type'] == vtype]  
 scatter = ax.scatter(  
 subset['Vehicle Speed[km/h]'],  
 subset['FCR'],  
 c=subset['Cluster'],  
 cmap='viridis',  
 s=50  
 )  
 # Map numeric vehicle type back to string for title  
 vtype\_str = {0: "ICE", 1: "HEV", 2: "EV", 3: "PHEV"}.get(vtype, str(vtype))  
 ax.set\_title(f'Vehicle Type: {vtype\_str}')  
 ax.set\_xlabel('Speed')  
 ax.set\_ylabel('FCR')  
 handles, labels = scatter.legend\_elements(prop="colors")  
 ax.legend(handles, [f"Cluster {i}" for i in range(len(handles))])  
  
fig.suptitle('KMeans Clusters: Speed vs FCR by Vehicle Type', fontsize=16)  
plt.tight\_layout()  
plt.show()



plot\_kmeans\_elbow(df\_combined\_sb)

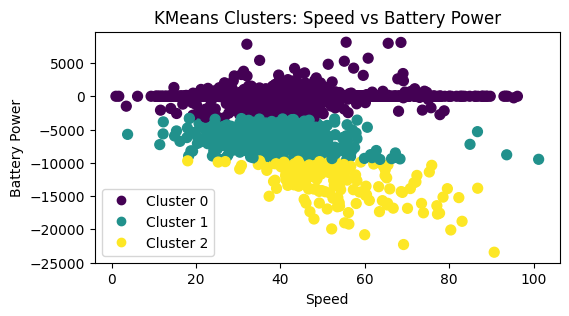
Suggested optimal K: 2



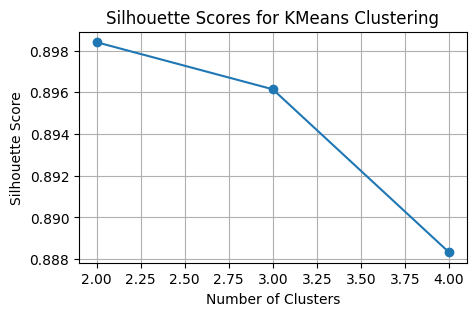
fit\_predict\_kmeans(df\_combined\_sb,3)

( Vehicle Speed[km/h] HV Battery Power[Watts] Cluster  
 0 44.100000 0.000000 0  
 1 24.263889 0.000000 0  
 2 36.552916 0.000000 0  
 3 30.005076 0.000000 0  
 4 45.045161 0.000000 0  
 ... ... ... ...  
 5045 44.174461 -11747.988095 2  
 5046 64.811233 -11926.910494 2  
 5047 38.361678 -12802.824627 2  
 5048 63.285435 -13596.112057 2  
 5049 58.763749 -14409.176136 2  
   
 [5050 rows x 3 columns],  
 array([0, 0, 0, ..., 2, 2, 2], shape=(5050,), dtype=int32),  
 KMeans(n\_clusters=3, random\_state=42))

'''  
- Creates a scatter plot of Vehicle Speed vs HV Battery Power, colored by KMeans cluster assignment.  
- Sets axis labels and plot title.  
- Adds a legend for the clusters.  
- Displays the plot.  
'''  
  
plt.figure(figsize=(6, 3))  
scatter = plt.scatter(  
 df\_combined\_sb['Vehicle Speed[km/h]'],  
 df\_combined\_sb['HV Battery Power[Watts]'],  
 c=df\_combined\_sb['Cluster'],  
 cmap='viridis',  
 s=50  
)  
plt.xlabel('Speed')  
plt.ylabel('Battery Power')  
plt.title('KMeans Clusters: Speed vs Battery Power')  
handles, labels = scatter.legend\_elements(prop="colors")  
plt.legend(handles, [f"Cluster {i}" for i in range(len(handles))])  
plt.show()



'''  
This code calculates the silhouette score for KMeans clustering on the features  
'Vehicle Speed[km/h]' and 'HV Battery Power[Watts]' from the dataframe df\_combined\_sb.  
- It iterates over a range of cluster numbers (2 to 10).  
- For each number of clusters, it fits a KMeans model and computes the silhouette score.  
- The silhouette score measures how well each data point fits within its cluster (higher is better).  
- Finally, it plots the silhouette scores against the number of clusters to help select the optimal cluster count.  
'''  
  
# Calculate silhouette score for each cluster  
from sklearn.metrics import silhouette\_score  
from sklearn.cluster import KMeans  
  
# Calculate silhouette score for each cluster  
silhouette\_scores = []  
for n\_clusters in range(2, 5):  
 kmeans = KMeans(n\_clusters=n\_clusters, random\_state=42)  
 kmeans.fit(df\_combined\_sb[['Vehicle Speed[km/h]', 'HV Battery Power[Watts]']])  
 labels = kmeans.labels\_  
 score = silhouette\_score(df\_combined\_sb[['Vehicle Speed[km/h]', 'HV Battery Power[Watts]']], labels)  
 silhouette\_scores.append((n\_clusters, score))  
   
# Plot silhouette scores  
plt.figure(figsize=(5, 3))  
plt.plot([score[0] for score in silhouette\_scores], [score[1] for score in silhouette\_scores], marker='o')  
plt.xlabel('Number of Clusters')  
plt.ylabel('Silhouette Score')  
plt.title('Silhouette Scores for KMeans Clustering')  
plt.grid(True)  
plt.show()

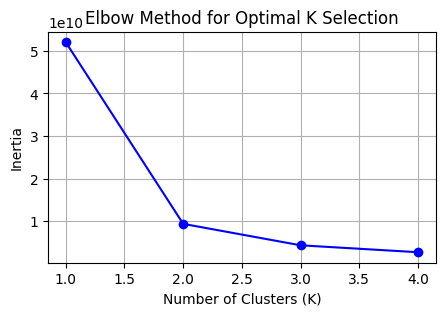


df\_combined\_sb = df\_combined[['Vehicle Type','Vehicle Speed[km/h]','HV Battery Power[Watts]']]  
# map vehicle type to 0,1,2,3  
df\_combined\_sb['Vehicle Type'] = df\_combined\_sb['Vehicle Type'].map({'ICE': 0, 'HEV': 1, 'EV': 2, 'PHEV': 3})  
  
df\_combined\_sb.head()

Vehicle Type Vehicle Speed[km/h] HV Battery Power[Watts]  
0 0 44.100000 0.0  
1 0 24.263889 0.0  
2 0 36.552916 0.0  
3 0 30.005076 0.0  
4 0 45.045161 0.0

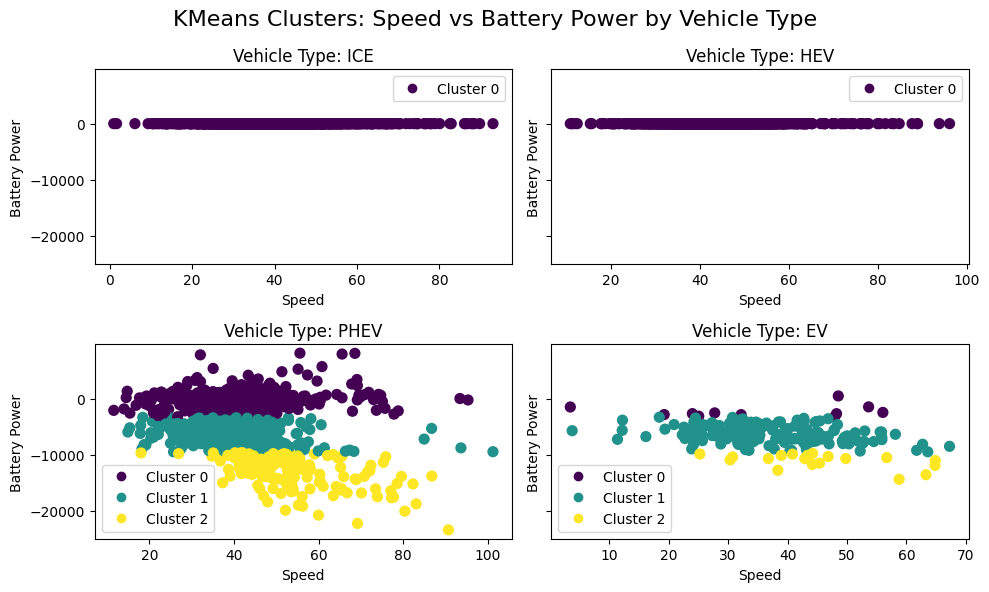
plot\_kmeans\_elbow(df\_combined\_sb)  
fit\_predict\_kmeans(df\_combined\_sb,3)

Suggested optimal K: 2



( Vehicle Type Vehicle Speed[km/h] HV Battery Power[Watts] Cluster  
 0 0 44.100000 0.000000 0  
 1 0 24.263889 0.000000 0  
 2 0 36.552916 0.000000 0  
 3 0 30.005076 0.000000 0  
 4 0 45.045161 0.000000 0  
 ... ... ... ... ...  
 5045 2 44.174461 -11747.988095 2  
 5046 2 64.811233 -11926.910494 2  
 5047 2 38.361678 -12802.824627 2  
 5048 2 63.285435 -13596.112057 2  
 5049 2 58.763749 -14409.176136 2  
   
 [5050 rows x 4 columns],  
 array([0, 0, 0, ..., 2, 2, 2], shape=(5050,), dtype=int32),  
 KMeans(n\_clusters=3, random\_state=42))

'''  
This code creates a 2x2 grid of scatter plots, one for each vehicle type, showing the relationship between 'Vehicle Speed[km/h]' and 'HV Battery Power[Watts]' for each type.   
- It iterates over unique vehicle types, selects the corresponding subset of data, and plots the points colored by their KMeans cluster assignment.  
- The numeric vehicle type is mapped back to a string for the plot title.  
- Each subplot includes a legend for the clusters.  
- The overall figure is titled and laid out neatly.  
'''  
  
import matplotlib.pyplot as plt  
  
vehicle\_types = df\_combined\_sf['Vehicle Type'].unique()  
  
fig, axes = plt.subplots(2, 2, figsize=(10, 6), sharey=True)  
axes = axes.flatten()  
  
for i, vtype in enumerate(vehicle\_types):  
 ax = axes[i]  
 subset = df\_combined\_sb[df\_combined\_sb['Vehicle Type'] == vtype]  
 scatter = ax.scatter(  
 subset['Vehicle Speed[km/h]'],  
 subset['HV Battery Power[Watts]'],  
 c=subset['Cluster'],  
 cmap='viridis',  
 s=50  
 )  
 # Map numeric vehicle type back to string for title  
 vtype\_str = {0: "ICE", 1: "HEV", 2: "EV", 3: "PHEV"}.get(vtype, str(vtype))  
 ax.set\_title(f'Vehicle Type: {vtype\_str}')  
 ax.set\_xlabel('Speed')  
 ax.set\_ylabel('Battery Power')  
 handles, labels = scatter.legend\_elements(prop="colors")  
 ax.legend(handles, [f"Cluster {i}" for i in range(len(handles))])  
  
fig.suptitle('KMeans Clusters: Speed vs Battery Power by Vehicle Type', fontsize=16)  
plt.tight\_layout()  
plt.show()

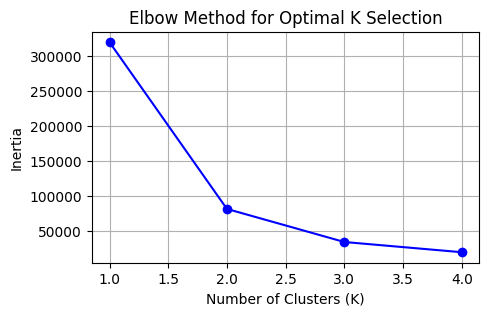


df\_combined\_sb

Vehicle Type Vehicle Speed[km/h] HV Battery Power[Watts] Cluster  
0 0 44.100000 0.000000 0  
1 0 24.263889 0.000000 0  
2 0 36.552916 0.000000 0  
3 0 30.005076 0.000000 0  
4 0 45.045161 0.000000 0  
... ... ... ... ...  
5045 2 44.174461 -11747.988095 2  
5046 2 64.811233 -11926.910494 2  
5047 2 38.361678 -12802.824627 2  
5048 2 63.285435 -13596.112057 2  
5049 2 58.763749 -14409.176136 2  
  
[5050 rows x 4 columns]

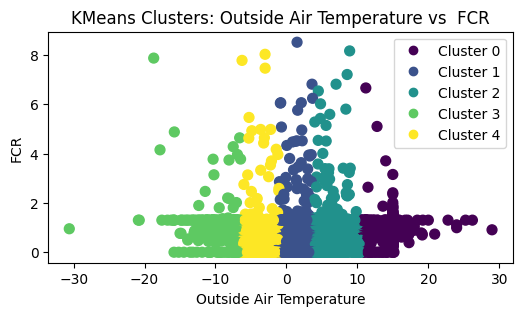
plot\_kmeans\_elbow(df\_combined\_of)  
fit\_predict\_kmeans(df\_combined\_of,5)

Suggested optimal K: 2



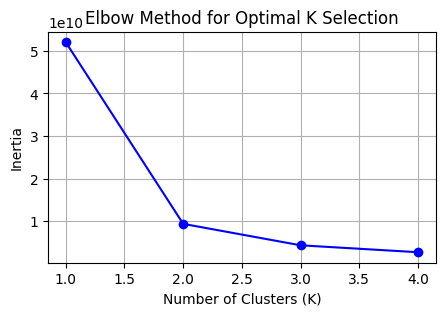
( OAT[DegC] FCR Cluster  
 0 15.000000 3.154674 0  
 1 15.000000 2.411606 0  
 2 15.000000 2.353453 0  
 3 15.000000 2.021139 0  
 4 -7.046392 2.015042 3  
 ... ... ... ...  
 5045 3.156085 0.000000 1  
 5046 6.500000 0.000000 2  
 5047 1.246269 0.000000 1  
 5048 -5.041553 0.000000 4  
 5049 -2.659091 0.000000 4  
   
 [5050 rows x 3 columns],  
 array([0, 0, 0, ..., 1, 4, 4], shape=(5050,), dtype=int32),  
 KMeans(n\_clusters=5, random\_state=42))

'''  
This code creates a scatter plot to visualize KMeans clustering results.  
- It plots 'OAT[DegC]' (Outside Air Temperature) vs 'FCR', coloring points by their cluster assignment.  
- The color map 'viridis' is used for cluster coloring.  
- The plot includes axis labels, a title, and a legend indicating cluster numbers.  
'''  
  
plt.figure(figsize=(6, 3))  
scatter = plt.scatter(  
 df\_combined\_of['OAT[DegC]'],  
 df\_combined\_of['FCR'],  
 c=df\_combined\_of['Cluster'],  
 cmap='viridis',  
 s=50  
)  
plt.xlabel('Outside Air Temperature')  
plt.ylabel('FCR')  
plt.title('KMeans Clusters: Outside Air Temperature vs FCR')  
handles, labels = scatter.legend\_elements(prop="colors")  
plt.legend(handles, [f"Cluster {i}" for i in range(len(handles))])  
plt.show()



plot\_kmeans\_elbow(df\_combined\_ob)  
fit\_predict\_kmeans(df\_combined\_ob,5)

Suggested optimal K: 2



( OAT[DegC] HV Battery Power[Watts] Cluster  
 0 15.000000 0.000000 0  
 1 15.000000 0.000000 0  
 2 15.000000 0.000000 0  
 3 15.000000 0.000000 0  
 4 -7.046392 0.000000 0  
 ... ... ... ...  
 5045 3.156085 -11747.988095 4  
 5046 6.500000 -11926.910494 4  
 5047 1.246269 -12802.824627 4  
 5048 -5.041553 -13596.112057 4  
 5049 -2.659091 -14409.176136 2  
   
 [5050 rows x 3 columns],  
 array([0, 0, 0, ..., 4, 4, 2], shape=(5050,), dtype=int32),  
 KMeans(n\_clusters=5, random\_state=42))

'''  
This code creates a scatter plot to visualize KMeans clustering results.  
- It plots 'OAT[DegC]' (Outside Air Temperature) vs 'HV Battery Power[Watts]', coloring points by their cluster assignment.  
- The color map 'viridis' is used for cluster coloring.  
- The plot includes axis labels, a title, and a legend indicating cluster numbers.  
'''  
  
plt.figure(figsize=(6, 3))  
scatter = plt.scatter(  
 df\_combined\_ob['OAT[DegC]'],  
 df\_combined\_ob['HV Battery Power[Watts]'],  
 c=df\_combined\_ob['Cluster'],  
 cmap='viridis',  
 s=50  
)  
plt.xlabel('Outside Air Temperature')  
plt.ylabel('Battery Power')  
plt.title('KMeans Clusters: Outside Air Temperature vs Battery Power')  
handles, labels = scatter.legend\_elements(prop="colors")  
plt.legend(handles, [f"Cluster {i}" for i in range(len(handles))])  
plt.show()

