Table of Contents

# 1 Static Data Load

import warnings  
warnings.filterwarnings('ignore')

import os  
print(os.getcwd())

g:\DIYguru\Data-Science-and-Engineering-Analytics\Projects\Main\_Project\_ML

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

'''  
  
# Import required libraries for Google Drive authentication  
from pydrive.auth import GoogleAuth  
from pydrive.drive import GoogleDrive  
  
def authenticate\_drive():  
 """  
 Authenticate and return a GoogleDrive instance.  
   
 This function handles Google Drive authentication using PyDrive.  
 It opens a browser window for user login and returns an authenticated  
 GoogleDrive instance that can be used to access files on Google Drive.  
   
 Returns:  
 GoogleDrive: An authenticated GoogleDrive instance  
 """  
 # Initialize GoogleAuth object for authentication  
 gauth = GoogleAuth()  
   
 # Perform local webserver authentication - opens browser for login  
 gauth.LocalWebserverAuth()  
   
 # Return authenticated GoogleDrive instance  
 return GoogleDrive(gauth)  
  
'''

'\n\n# Import required libraries for Google Drive authentication\nfrom pydrive.auth import GoogleAuth\nfrom pydrive.drive import GoogleDrive\n\ndef authenticate\_drive():\n """\n Authenticate and return a GoogleDrive instance.\n \n This function handles Google Drive authentication using PyDrive.\n It opens a browser window for user login and returns an authenticated\n GoogleDrive instance that can be used to access files on Google Drive.\n \n Returns:\n GoogleDrive: An authenticated GoogleDrive instance\n """\n # Initialize GoogleAuth object for authentication\n gauth = GoogleAuth()\n \n # Perform local webserver authentication - opens browser for login\n gauth.LocalWebserverAuth()\n \n # Return authenticated GoogleDrive instance\n return GoogleDrive(gauth)\n\n'

'''  
  
This cell defines a function to read Excel files from Google Drive.  
The function takes an authenticated GoogleDrive instance and a file\_id as parameters.  
It downloads the Excel file to a temporary location and reads it into a pandas DataFrame.  
This enables accessing Excel files stored in Google Drive programmatically.  
  
  
import pandas as pd  
  
def read\_excel\_from\_drive(drive, file\_id):  
 """  
 Reads an Excel file from Google Drive using its file\_id.  
   
 Args:  
 drive: Authenticated GoogleDrive instance  
 file\_id: File ID of the Excel file in Google Drive  
   
 Returns:  
 DataFrame containing the Excel data  
 """  
 file = drive.CreateFile({'id': file\_id})  
 file.GetContentFile('temp\_excel\_file.xlsx')  
 df = pd.read\_excel('temp\_excel\_file.xlsx')  
 return df  
  
'''

'\n\nThis cell defines a function to read Excel files from Google Drive.\nThe function takes an authenticated GoogleDrive instance and a file\_id as parameters.\nIt downloads the Excel file to a temporary location and reads it into a pandas DataFrame.\nThis enables accessing Excel files stored in Google Drive programmatically.\n\n\nimport pandas as pd\n\ndef read\_excel\_from\_drive(drive, file\_id):\n """\n Reads an Excel file from Google Drive using its file\_id.\n \n Args:\n drive: Authenticated GoogleDrive instance\n file\_id: File ID of the Excel file in Google Drive\n \n Returns:\n DataFrame containing the Excel data\n """\n file = drive.CreateFile({\'id\': file\_id})\n file.GetContentFile(\'temp\_excel\_file.xlsx\')\n df = pd.read\_excel(\'temp\_excel\_file.xlsx\')\n return df\n\n'

'''  
This cell loads AWS credentials from a .env file using python-dotenv,  
creates a boto3 S3 client with those credentials,  
and lists all S3 buckets in the account.  
'''  
  
import os  
from dotenv import load\_dotenv  
import boto3  
  
# Load environment variables from .env  
load\_dotenv()  
  
# Create boto3 client using loaded environment variables  
s3 = boto3.client("s3",  
 aws\_access\_key\_id=os.getenv("AWS\_ACCESS\_KEY\_ID"),  
 aws\_secret\_access\_key=os.getenv("AWS\_SECRET\_ACCESS\_KEY"),  
 region\_name=os.getenv("AWS\_DEFAULT\_REGION")  
)  
  
# Example: list buckets  
buckets = s3.list\_buckets()  
print("Your S3 Buckets:")  
for bucket in buckets['Buckets']:  
 print(f" - {bucket['Name']}")

Your S3 Buckets:  
 - s3aravindh973515031797

'''  
  
import pandas as pd  
from io import BytesIO  
  
def read\_excel\_from\_s3(bucket\_name, object\_key):  
 """  
 Reads an Excel file from an AWS S3 bucket using the global s3 client.  
  
 Args:  
 bucket\_name: Name of the S3 bucket.  
 object\_key: Key (path) to the Excel file in the S3 bucket.  
  
 Returns:  
 DataFrame containing the Excel data.  
 """  
 response = s3.get\_object(Bucket=bucket\_name, Key=object\_key)  
 file\_content = response['Body'].read()  
 df = pd.read\_excel(BytesIO(file\_content))  
 return df  
  
'''

'\n\nimport pandas as pd\nfrom io import BytesIO\n\ndef read\_excel\_from\_s3(bucket\_name, object\_key):\n """\n Reads an Excel file from an AWS S3 bucket using the global s3 client.\n\n Args:\n bucket\_name: Name of the S3 bucket.\n object\_key: Key (path) to the Excel file in the S3 bucket.\n\n Returns:\n DataFrame containing the Excel data.\n """\n response = s3.get\_object(Bucket=bucket\_name, Key=object\_key)\n file\_content = response[\'Body\'].read()\n df = pd.read\_excel(BytesIO(file\_content))\n return df\n\n'

'''  
# Step 1: Authenticate  
drive = authenticate\_drive()  
  
# Step 2: Provide File and Folder IDs  
ICE\_HEV = '1nIVzW4czIBAOczy9DExM3VEVyYGBXNJe' # Replace with your Excel file ID  
PHEV\_EV = '1L-rPZ-OrASQDw4m-onNPZFSTNAkbEe8H' # Replace with your Excel file ID  
  
# Step 3: Read Excel  
df\_ICE\_HEV = read\_excel\_from\_drive(drive, ICE\_HEV)  
print("Excel File Contents:\n", df\_ICE\_HEV.head())  
  
df\_PHEV\_EV = read\_excel\_from\_drive(drive, PHEV\_EV)  
print("Excel File Contents:\n", df\_PHEV\_EV.head())  
'''

'\n# Step 1: Authenticate\ndrive = authenticate\_drive()\n\n# Step 2: Provide File and Folder IDs\nICE\_HEV = \'1nIVzW4czIBAOczy9DExM3VEVyYGBXNJe\' # Replace with your Excel file ID\nPHEV\_EV = \'1L-rPZ-OrASQDw4m-onNPZFSTNAkbEe8H\' # Replace with your Excel file ID\n\n# Step 3: Read Excel\ndf\_ICE\_HEV = read\_excel\_from\_drive(drive, ICE\_HEV)\nprint("Excel File Contents:\n", df\_ICE\_HEV.head())\n\ndf\_PHEV\_EV = read\_excel\_from\_drive(drive, PHEV\_EV)\nprint("Excel File Contents:\n", df\_PHEV\_EV.head())\n'

bucket\_name = 's3aravindh973515031797'  
  
response = s3.list\_objects\_v2(Bucket=bucket\_name)  
for item in response.get("Contents", []):  
 print(item["Key"])

Cleaned up VED Source Data/  
Cleaned up VED Source Data/df\_ICE\_HEV.parquet  
Cleaned up VED Source Data/df\_PHEV\_EV.parquet  
Cleaned up VED Source Data/df\_VED.parquet  
Cleaned up VED Source Data/df\_combined.parquet  
Cleaned up VED Source Data/df\_dynamic\_sample.parquet  
Cleaned up VED Source Data/df\_static.parquet  
DIYguru ML Source Data/  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_171101\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_171108\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_171115\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_171122\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_171129\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_171206\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_171213\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_171220\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_171227\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_180103\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_180110\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_180117\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_180124\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_180131\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_180207\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_180214\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_180221\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_180228\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_180307\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_180314\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_180321\_week.csv  
DIYguru ML Source Data/VED\_DynamicData\_Part1/VED\_180328\_week.csv  
DIYguru ML Source Data/VED\_Static\_Data\_ICE&HEV.xlsx  
DIYguru ML Source Data/VED\_Static\_Data\_PHEV&EV.xlsx

'''  
ICE\_HEV = 'DIYguru ML Source Data/VED\_Static\_Data\_ICE&HEV.xlsx'  
PHEV\_EV = 'DIYguru ML Source Data/VED\_Static\_Data\_PHEV&EV.xlsx'  
  
df\_ICE\_HEV = read\_excel\_from\_s3(bucket\_name, ICE\_HEV)  
print("Excel File Contents:\n", df\_ICE\_HEV.head())  
  
df\_PHEV\_EV = read\_excel\_from\_s3(bucket\_name, PHEV\_EV)  
print("Excel File Contents:\n", df\_PHEV\_EV.head())  
'''

'\nICE\_HEV = \'DIYguru ML Source Data/VED\_Static\_Data\_ICE&HEV.xlsx\'\nPHEV\_EV = \'DIYguru ML Source Data/VED\_Static\_Data\_PHEV&EV.xlsx\'\n\ndf\_ICE\_HEV = read\_excel\_from\_s3(bucket\_name, ICE\_HEV)\nprint("Excel File Contents:\n", df\_ICE\_HEV.head())\n\ndf\_PHEV\_EV = read\_excel\_from\_s3(bucket\_name, PHEV\_EV)\nprint("Excel File Contents:\n", df\_PHEV\_EV.head())\n'

'''  
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]

import pandas as pd  
import numpy as np  
import os  
  
'''  
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)

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

'''  
def push\_df\_to\_s3(df, object\_key):  
 """  
 Pushes a pandas DataFrame to a predefined S3 bucket as a CSV file.  
  
 Args:  
 df (pd.DataFrame): DataFrame to upload.  
 object\_key (str): S3 object key (path/filename.csv).  
 """  
 from io import BytesIO  
 csv\_buffer = BytesIO()  
 df.to\_csv(csv\_buffer, index=False)  
 csv\_buffer.seek(0)  
 s3.upload\_fileobj(csv\_buffer, bucket\_name, object\_key)  
 print(f"DataFrame uploaded to s3://{bucket\_name}/{object\_key}")  
  
# Example usage:  
# push\_df\_to\_s3(df\_static, 'path/to/df\_static.csv')  
'''

'\ndef push\_df\_to\_s3(df, object\_key):\n """\n Pushes a pandas DataFrame to a predefined S3 bucket as a CSV file.\n\n Args:\n df (pd.DataFrame): DataFrame to upload.\n object\_key (str): S3 object key (path/filename.csv).\n """\n from io import BytesIO\n csv\_buffer = BytesIO()\n df.to\_csv(csv\_buffer, index=False)\n csv\_buffer.seek(0)\n s3.upload\_fileobj(csv\_buffer, bucket\_name, object\_key)\n print(f"DataFrame uploaded to s3://{bucket\_name}/{object\_key}")\n\n# Example usage:\n# push\_df\_to\_s3(df\_static, \'path/to/df\_static.csv\')\n'

'''  
push\_df\_to\_s3(df\_static, 'Cleaned up VED Source Data/df\_static.csv')  
'''

"\npush\_df\_to\_s3(df\_static, 'Cleaned up VED Source Data/df\_static.csv')\n"

def push\_df\_to\_s3\_parquet(df, object\_key):  
 """  
 Pushes a pandas DataFrame to a predefined S3 bucket as a Parquet file.  
  
 Args:  
 df (pd.DataFrame): DataFrame to upload.  
 object\_key (str): S3 object key (path/filename.parquet).  
 """  
 from io import BytesIO  
 parquet\_buffer = BytesIO()  
 df.to\_parquet(parquet\_buffer, index=False)  
 parquet\_buffer.seek(0)  
 s3.upload\_fileobj(parquet\_buffer, bucket\_name, object\_key)  
 print(f"DataFrame uploaded to s3://{bucket\_name}/{object\_key}")  
  
# Example usage:  
# push\_df\_to\_s3\_parquet(df\_static, 'path/to/df\_static.parquet')

push\_df\_to\_s3\_parquet(df\_ICE\_HEV, 'Cleaned up VED Source Data/df\_ICE\_HEV.parquet')  
push\_df\_to\_s3\_parquet(df\_PHEV\_EV, 'Cleaned up VED Source Data/df\_PHEV\_EV.parquet')  
push\_df\_to\_s3\_parquet(df\_static, 'Cleaned up VED Source Data/df\_static.parquet')

DataFrame uploaded to s3://s3aravindh973515031797/Cleaned up VED Source Data/df\_ICE\_HEV.parquet  
DataFrame uploaded to s3://s3aravindh973515031797/Cleaned up VED Source Data/df\_PHEV\_EV.parquet  
DataFrame uploaded to s3://s3aravindh973515031797/Cleaned up VED Source Data/df\_static.parquet

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

# 2 Dynamic Data Load

'''  
Loads all CSV files from a specified directory, randomly samples 50% of them, reads each sampled file into a DataFrame, and concatenates them into a single DataFrame.  
'''  
  
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')]  
 np.random.seed(42)  
 sampled\_files = np.random.choice(all\_files, size=int(len(all\_files) \* 0.5), replace=False)  
 df\_list = []  
   
 for file in sampled\_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 using the above function  
Loaded VED\_DynamicData\_Part2 into dataframe df\_part2 using the above function  
'''  
  
df\_part1 = load\_csv\_files\_from\_directory("G:\\DIYguru\\Notes and Sample Data\\VED-master\\Data\\VED\_DynamicData\_Part1")  
#df\_part2 = load\_csv\_files\_from\_directory("G:\\DIYguru\\Notes and Sample Data\\VED-master\\Data\\VED\_DynamicData\_Part2")

'''  
  
import random  
import pandas as pd  
from io import StringIO  
  
def read\_and\_concat\_random\_csvs\_from\_drive\_folder(drive, folder\_id, sample\_frac=0.5):  
 """  
 Reads a random fraction of CSV files directly from a Google Drive folder   
 and concatenates them into a single DataFrame.  
   
 Args:  
 drive: Authenticated GoogleDrive instance  
 folder\_id: Folder ID in Google Drive  
 sample\_frac: Fraction of CSV files to read (default is 0.5)  
   
 Returns:  
 A single pandas DataFrame combining all sampled CSV files  
 """  
 file\_list = drive.ListFile({'q': f"'{folder\_id}' in parents and trashed=false"}).GetList()  
  
 csv\_files = [file for file in file\_list if file['title'].endswith('.csv')]  
  
 if not csv\_files:  
 return pd.DataFrame() # Return empty DataFrame if no CSVs found  
  
 sample\_size = max(1, int(len(csv\_files) \* sample\_frac))  
 sampled\_files = random.sample(csv\_files, sample\_size)  
  
 dataframes = []  
 for file in sampled\_files:  
 file\_content = file.GetContentString()  
 df = pd.read\_csv(StringIO(file\_content))  
 df['source\_file'] = file['title'] # Optional: add filename for tracking  
 dataframes.append(df)  
  
 combined\_df = pd.concat(dataframes, ignore\_index=True)  
 return combined\_df  
  
'''

'\n\nimport random\nimport pandas as pd\nfrom io import StringIO\n\ndef read\_and\_concat\_random\_csvs\_from\_drive\_folder(drive, folder\_id, sample\_frac=0.5):\n """\n Reads a random fraction of CSV files directly from a Google Drive folder \n and concatenates them into a single DataFrame.\n \n Args:\n drive: Authenticated GoogleDrive instance\n folder\_id: Folder ID in Google Drive\n sample\_frac: Fraction of CSV files to read (default is 0.5)\n \n Returns:\n A single pandas DataFrame combining all sampled CSV files\n """\n file\_list = drive.ListFile({\'q\': f"\'{folder\_id}\' in parents and trashed=false"}).GetList()\n\n csv\_files = [file for file in file\_list if file[\'title\'].endswith(\'.csv\')]\n\n if not csv\_files:\n return pd.DataFrame() # Return empty DataFrame if no CSVs found\n\n sample\_size = max(1, int(len(csv\_files) \* sample\_frac))\n sampled\_files = random.sample(csv\_files, sample\_size)\n\n dataframes = []\n for file in sampled\_files:\n file\_content = file.GetContentString()\n df = pd.read\_csv(StringIO(file\_content))\n df[\'source\_file\'] = file[\'title\'] # Optional: add filename for tracking\n dataframes.append(df)\n\n combined\_df = pd.concat(dataframes, ignore\_index=True)\n return combined\_df\n\n'

'''  
  
import random  
import pandas as pd  
import boto3  
from io import StringIO  
  
def read\_and\_concat\_random\_csvs\_from\_s3(bucket\_name, folder\_key, sample\_frac=0.5):  
 """  
 Reads a random fraction of CSV files directly from an AWS S3 bucket folder   
 and concatenates them into a single DataFrame.  
  
 Args:  
 bucket\_name: Name of the S3 bucket  
 folder\_key: S3 folder key (prefix) where CSV files are stored  
 sample\_frac: Fraction of CSV files to read (default is 0.5)  
  
 Returns:  
 A single pandas DataFrame combining all sampled CSV files  
 """  
 # Assume s3 variable is a boto3 S3 client or resource, already authenticated  
 # Example: s3 = boto3.client('s3') or s3 = boto3.resource('s3')  
 # Here, we use s3 as a boto3 client  
 global s3 # s3 should be defined elsewhere in the notebook/environment  
 random.seed(42)  
  
 # List all objects in the specified S3 folder  
 response = s3.list\_objects\_v2(Bucket=bucket\_name, Prefix=folder\_key)  
 if 'Contents' not in response:  
 return pd.DataFrame() # Return empty DataFrame if no files found  
  
 # Filter for CSV files  
 csv\_files = [obj['Key'] for obj in response['Contents'] if obj['Key'].endswith('.csv')]  
  
 if not csv\_files:  
 return pd.DataFrame() # Return empty DataFrame if no CSVs found  
  
 sample\_size = max(1, int(len(csv\_files) \* sample\_frac))  
 sampled\_files = random.sample(csv\_files, sample\_size)  
  
 dataframes = []  
 for key in sampled\_files:  
 obj = s3.get\_object(Bucket=bucket\_name, Key=key)  
 file\_content = obj['Body'].read().decode('utf-8')  
 df = pd.read\_csv(StringIO(file\_content))  
 df['source\_file'] = key # Optional: add filename for tracking  
 dataframes.append(df)  
  
 combined\_df = pd.concat(dataframes, ignore\_index=True)  
 return combined\_df  
  
'''

'\n\nimport random\nimport pandas as pd\nimport boto3\nfrom io import StringIO\n\ndef read\_and\_concat\_random\_csvs\_from\_s3(bucket\_name, folder\_key, sample\_frac=0.5):\n """\n Reads a random fraction of CSV files directly from an AWS S3 bucket folder \n and concatenates them into a single DataFrame.\n\n Args:\n bucket\_name: Name of the S3 bucket\n folder\_key: S3 folder key (prefix) where CSV files are stored\n sample\_frac: Fraction of CSV files to read (default is 0.5)\n\n Returns:\n A single pandas DataFrame combining all sampled CSV files\n """\n # Assume s3 variable is a boto3 S3 client or resource, already authenticated\n # Example: s3 = boto3.client(\'s3\') or s3 = boto3.resource(\'s3\')\n # Here, we use s3 as a boto3 client\n global s3 # s3 should be defined elsewhere in the notebook/environment\n random.seed(42)\n\n # List all objects in the specified S3 folder\n response = s3.list\_objects\_v2(Bucket=bucket\_name, Prefix=folder\_key)\n if \'Contents\' not in response:\n return pd.DataFrame() # Return empty DataFrame if no files found\n\n # Filter for CSV files\n csv\_files = [obj[\'Key\'] for obj in response[\'Contents\'] if obj[\'Key\'].endswith(\'.csv\')]\n\n if not csv\_files:\n return pd.DataFrame() # Return empty DataFrame if no CSVs found\n\n sample\_size = max(1, int(len(csv\_files) \* sample\_frac))\n sampled\_files = random.sample(csv\_files, sample\_size)\n\n dataframes = []\n for key in sampled\_files:\n obj = s3.get\_object(Bucket=bucket\_name, Key=key)\n file\_content = obj[\'Body\'].read().decode(\'utf-8\')\n df = pd.read\_csv(StringIO(file\_content))\n df[\'source\_file\'] = key # Optional: add filename for tracking\n dataframes.append(df)\n\n combined\_df = pd.concat(dataframes, ignore\_index=True)\n return combined\_df\n\n'

# Recursively list all folders and subfolders in the S3 bucket  
  
def list\_all\_s3\_folders(bucket\_name, prefix=''):  
 """  
 Recursively lists all folders and subfolders in an S3 bucket.  
 Returns a list of folder prefixes.  
 """  
 folders = []  
 paginator = s3.get\_paginator('list\_objects\_v2')  
 for page in paginator.paginate(Bucket=bucket\_name, Prefix=prefix, Delimiter='/'):  
 if 'CommonPrefixes' in page:  
 for cp in page['CommonPrefixes']:  
 folder\_prefix = cp['Prefix']  
 folders.append(folder\_prefix)  
 # Recursively get subfolders  
 folders.extend(list\_all\_s3\_folders(bucket\_name, prefix=folder\_prefix))  
 return folders  
  
all\_folders = list\_all\_s3\_folders(bucket\_name)  
if all\_folders:  
 print("📁 All folders and subfolders:")  
 for folder in all\_folders:  
 print(f" - {folder}")  
else:  
 print("No folders found.")

📁 All folders and subfolders:  
 - Cleaned up VED Source Data/  
 - DIYguru ML Source Data/  
 - DIYguru ML Source Data/VED\_DynamicData\_Part1/

'''  
  
part1 = 'DIYguru ML Source Data/VED\_DynamicData\_Part1/'   
  
df\_part1 = read\_and\_concat\_random\_csvs\_from\_s3(bucket\_name, part1, sample\_frac=0.5)  
  
# Print each file name and first few rows  
for filename, df in df\_part1.items():  
 print(f"\n--- {filename} ---\n", df.head())  
  
  
'''

'\n\npart1 = \'DIYguru ML Source Data/VED\_DynamicData\_Part1/\' \n\ndf\_part1 = read\_and\_concat\_random\_csvs\_from\_s3(bucket\_name, part1, sample\_frac=0.5)\n\n# Print each file name and first few rows\nfor filename, df in df\_part1.items():\n print(f"\n--- {filename} ---\n", df.head())\n\n\n'

'''  
  
part1 = '13K9WanXU7lOd-nWzztKoCF4kH7LC5789' # Replace with your folder ID  
  
# Step 4: Read Random Sample of CSVs  
df\_part1 = read\_and\_concat\_random\_csvs\_from\_drive\_folder(drive, part1)  
  
# Print each file name and first few rows  
for filename, df in df\_part1.items():  
 print(f"\n--- {filename} ---\n", df.head())  
  
'''

'\n\npart1 = \'13K9WanXU7lOd-nWzztKoCF4kH7LC5789\' # Replace with your folder ID\n\n# Step 4: Read Random Sample of CSVs\ndf\_part1 = read\_and\_concat\_random\_csvs\_from\_drive\_folder(drive, part1)\n\n# Print each file name and first few rows\nfor filename, df in df\_part1.items():\n print(f"\n--- {filename} ---\n", df.head())\n\n'

df\_part1.columns

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[%]'],  
 dtype='object')

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

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5131987 entries, 0 to 5131986  
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: 861.4 MB

'''  
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 ... 126.83646968 126.8464274  
 126.04467598]  
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  
 311 383 404 536 563 571 202 252 288 349 366 406 409 436 458 474 475 490  
 591 602 603 605 5 142 169 381 416 435 448 470 604 135 206 437 467 542  
 607 609 123 274 333 598 167 214 286 313 466 120 429 503 159 201 539 534  
 303 447 600 251 2 162 476 164 487 526 610 137 191 285 329 386 129 414  
 492 577 211 148 270 275 348 376 461 599 255 380 325 328 397 527 360 567  
 618 580 545 616 172 247 558 297 357 379 306 541 9]  
Unique values in column 'Trip':  
[ 706 707 1558 ... 2257 1502 1901]  
Unique values in column 'Timestamp(ms)':  
[ 0 200 1100 ... 8377200 3016500 3027200]  
Unique values in column 'Latitude[deg]':  
[42.27755833 42.27825528 42.2790125 ... 42.26687 42.26242389  
 42.259955 ]  
Unique values in column 'Longitude[deg]':  
[-83.69874972 -83.69880306 -83.69890111 ... -83.71665333 -83.71606639  
 -83.7147225 ]  
Unique values in column 'Vehicle Speed[km/h]':  
[ 40. 45. 47. ... 120.15625 115.28125 85.984375]  
Unique values in column 'MAF[g/sec]':  
[22.12999916 6.1500001 21.44000053 ... 72.54000092 73.88999939  
 74.79000092]  
Unique values in column 'Engine RPM[RPM]':  
[2285. 2744. 1982. ... 5357. 4427. 4429.]  
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  
 1.33333328e+02 1.22352943e+02 1.66274506e+02 1.16470589e+02  
 1.34117645e+02 1.43529419e+02 1.17254906e+02 1.38823532e+02  
 1.04313728e+02 1.96078432e+00 1.28576475e+04 1.72156860e+02  
 1.08627449e+02 1.28235291e+02 1.10588234e+02 1.17647064e+00  
 1.15686279e+02 1.05490196e+02 1.07450981e+02 1.13725494e+02  
 1.10980392e+02 1.10196083e+02 2.41568634e+02 1.34509811e+02  
 1.32941177e+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 1.16862747e+02 1.14901962e+02  
 1.39215683e+02 1.41568634e+02 1.32156860e+02 1.55686279e+02  
 1.58039215e+02 1.30588242e+02 2.30588242e+02 1.73435293e+04  
 1.86274506e+02 1.52549026e+02 1.18431374e+02 1.75686279e+02  
 1.65639219e+04 1.84313736e+02 1.74117645e+02 1.63529419e+02  
 1.59215683e+02 1.48627457e+02 1.57647064e+02 1.20392159e+02  
 1.52156860e+02 1.74509811e+02 1.41960785e+02 1.29803925e+02  
 1.33725494e+02 1.44313721e+02 1.49019608e+02 1.60392166e+02  
 1.18039215e+02 1.20784317e+02 1.46274506e+02 1.23529411e+02  
 1.69803925e+02 1.56862748e+00 2.06274506e+02 1.85882355e+02  
 1.65098038e+02 1.30980392e+02 1.36141182e+04 1.40000000e+02  
 1.70980392e+02 1.60345098e+04 1.13333336e+02 1.80784317e+02  
 1.38431381e+02 1.70588242e+02 1.74901962e+02 1.65882355e+02  
 2.16470596e+02 1.61176468e+02 1.54509811e+02 1.40392166e+02  
 1.50196075e+02 1.21960785e+02 1.51372559e+02 1.46666672e+02  
 1.38039215e+02 1.47843140e+02 1.49803925e+02 1.45882355e+02  
 1.32549026e+02 1.52941177e+02 2.03529419e+02 1.59607849e+02  
 1.43137253e+02 1.48235291e+02 1.73333344e+02 1.90196075e+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. -5.5 -5. -6. -7. -1. 0. -10. -3. -4.  
 -2. 1. -3.5 -2.5 -3.25 -3.75 -4.75 -5.25 -4.5 -5.75  
 -6.25 -6.5 -6.75 -0.75 -1.75 -1.5 -2.25 -1.25 -2.75 -8.  
 0.5 -0.5 -7.5 -4.25 -9. -11. -8.25 -8.5 -9.75 -9.25  
 -12. -13. -7.75 -7.25 -10.25 -10.5 0.25 1.25 0.75 21.  
 -9.5 -11.75 -10.75 -14. -17. -16. -12.5 -12.75 -12.25 -13.5  
 -0.25 -11.5 -18. -11.25 -17.5 -17.75 -17.25 -18.5 -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 -8.75 -20.5  
 -21. 33. -40. -39. 17.5 18.25 18.5 18.75 15.75 16.25  
 16.75 14.25 17.75 -38. -37. -36. 34. ]  
Unique values in column 'Fuel Rate[L/hr]':  
[ nan 0. 6.69228172 ... 4.62211037 3.82075405 1.49300373]  
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 6.27999973 6.35999966 6.11999989 6. 5.96000004  
 5.87999964 5.51999998 5.35999966 5.44000006 5.4000001 5.27999973  
 5.31999969 0.12 4.75999975 4.19999981 4.83999968 5.19999981]  
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. 600. 1200. 1100. 1150.  
 1400. 50. 1800. 80. 1300. 1950. 1900. 1850. 1250. 2500. 2050. 2350.  
 1700. 1750. 520. 40. 1500.]  
Unique values in column 'Heater Power[Watts]':  
[ nan 2250. 2000. 1750. 1500. 1250. 1000. 750. 500. 250. 0. 4000.  
 3500. 3750. 3000. 3250. 2750. 2500.]  
Unique values in column 'HV Battery Current[A]':  
[ nan -21.5 23.5 ... -165.30000305 -155.55000305  
 -174.30000305]  
Unique values in column 'HV Battery SOC[%]':  
[ nan 96.34146881 95.97561646 ... 63.2911377 56.68354034  
 44.93670654]  
Unique values in column 'HV Battery Voltage[V]':  
[ nan 386. 390.5 384.5 387.  
 382.5 389.5 387.5 378.5 383.5  
 389. 388.5 381.5 291.5 300.  
 294.5 292.5 287.5 296.5 293.  
 296. 289. 301.5 295. 295.5  
 284. 293.5 297. 279. 290.  
 292. 304. 294. 298.5 297.5  
 291. 307.5 285. 300.5 288.  
 309.5 299. 299.5 301. 302.  
 274. 306. 302.5 309. 298.  
 308. 287. 286. 286.5 290.5  
 282.5 283. 303.5 288.5 345.  
 348. 346. 347. 343. 352.  
 359. 350. 351. 353. 356.  
 338. 349. 360. 339. 341.  
 337. 342. 344. 330. 332.  
 333. 340. 329. 331. 326.  
 334. 336. 357. 335. 358.  
 359.5 361.25 361.75 357.375 360.5  
 368.125 362.25 356.375 358.875 359.25  
 371.875 356.875 354.75 361.375 361.625  
 352.25 354.625 366.125 352.375 357.875  
 357.5 358.25 360.75 361. 371.25  
 362. 361.5 357.625 367.5 360.375  
 357.75 360.25 361.125 359.875 365.25  
 361.875 354.125 355.875 365.75 360.875  
 360.625 352.75 354. 354.5 363.75  
 359.625 357.125 355.125 359.375 358.75  
 358.5 362.625 358.125 355.25 362.875  
 354.375 355. 357.25 352.875 354.875  
 359.125 350.25 358.625 358.375 356.25  
 355.75 356.75 356.125 355.5 356.625  
 355.625 356.5 352.125 353.875 362.75  
 353.75 351.5 352.625 351.375 349.5  
 363.375 348.5 338.75 339.625 332.5  
 341.75 344.625 347.75 328.375 348.125  
 345.25 348.75 338.375 338.25 347.125  
 339.375 346.75 366.375 369.25 374.75  
 375.125 370.5 370.625 367. 359.75  
 362.5 365.875 366.875 367.375 368.375  
 363. 364.25 369. 378.375 366.625  
 367.25 367.625 368. 367.875 368.625  
 365. 353.25 364.625 366.5 367.125  
 354.25 353.625 347.375 362.125 362.375  
 349.375 364.5 353.125 351.75 350.875  
 380. 371.5 382. 379.5 377.5  
 376.5 375.5 371. 374.5 379.  
 373.5 374. 373. 385. 378.  
 381. 375. 377. 384. 380.5  
 383. 331.5 322.5 326.5 316.  
 324.5 320. 324. 327.5 328.5  
 325. 322. 323.5 321.5 325.5  
 316.5 318.5 314.5 315.5 310.  
 318. 321. 323. 313. 319.5  
 334.5 310.5 307. 317.5 311.  
 305.5 319. 315. 313.5 314.  
 312. 308.5 305. 311.5 304.5  
 317. 200. 195. 198.5 201.5  
 199.5 207. 208.5 209.5 215.  
 198. 197. 209. 213.5 208.  
 196.5 205.5 201. 200.5 199.  
 206.5 207.5 217.5 195.5 206.  
 218.5 193.5 191.5 289.5 284.5  
 303. 285.5 306.5 312.5 283.5  
 278. 280.5 383.625 385.375 388.375  
 392.375 386.75 384.25 372. 368.75  
 373.625 377.125 381.875 395.375 376.125  
 377.75 376.875 378.625 382.25 385.75  
 372.125 380.875 384.75 388. 388.75  
 383.875 379.125 192. 202. 216.5  
 194.5 190. 194. 197.5 193.  
 188. 196. 202.5 210.5 203.5  
 211. 187. 212. 186. 190.5  
 191. 204.5 192.5 210. 204.  
 188.5 189.5 181. 184. 205.  
 186.5 214.5 203. 211.5 212.5  
 189. 338.5 337.5 333.5 336.5  
 330.5 327. 335.5 342.5 341.5  
 345.5 329.5 328. 183. 215.5  
 213. 216. 217. 339.5 320.5  
 371.375 380.25 369.75 363.5 364.125  
 370.125 366.25 371.625 371.125 364.75  
 370.375 370.25 370.875 366. 363.125  
 363.625 363.25 360.125 185. 184.5  
 183.5 376. 370. 385.5 372.5  
 368.5 369.5 365.5 364. 282.  
 281.5 281. 278.5 277. 270.5  
 276.5 274.5 385.875 393.75 386.125  
 380.75 381.25 382.375 385.25 386.625  
 386.875 387.625 382.75 381.75 384.875  
 383.375 386.25 388.25 385.125 390.125  
 387.875 386.375 386.5 384.375 379.25  
 383.125 382.125 381.625 376.375 379.875  
 380.125 379.375 380.625 379.75 380.375  
 379.625 374.875 373.75 374.625 377.375  
 375.25 372.25 369.625 367.75 370.75  
 387.375 376.625 378.125 368.875 374.25  
 279.5 280. 340.5 273. 352.5  
 353.5 350.75 351.25 349.875 355.375  
 350.125 349.625 349.125 349.75 350.5  
 344.5 346.875 348.25 347.875 348.875  
 351.125 345.125 346.625 348.375 344.125  
 345.375 345.625 346.125 345.875 349.25  
 351.625 393. 392.5 391. 391.5  
 390. 392. 393.5 394. 347.5  
 362.95999146 350.47998047 351.51998901 360.87997437 360.35998535  
 357.23999023 356.72000122 372.83999634 369.19998169 371.79998779  
 370.75997925 367.11999512 362.43997192 366.07998657 366.59997559  
 365.55999756 365.03997803 364.51998901 361.91998291 355.15997314  
 353.07998657 354.63998413 349.44000244 354.11999512 357.75997925  
 368.67999268 352.55999756 363.47998047 358.79998779 355.67999268  
 356.19998169 369.72000122 359.83999634 385.625 382.875  
 375.625 381.375 389.625 388.125 374.375  
 382.625 390.375 394.625 377.625 214.  
 375.875 374.125 363.875 344.375 353.375  
 350.375 343.625 369.125 368.25 369.875  
 372.875 372.375 365.375 366.75 369.375  
 373.25 365.125 364.875 361.3999939 358.27999878  
 347.35998535 349.95999146 347.87997437 353.59997559 341.63998413  
 352.03997803 344.75997925 346.83999634 348.91998291 348.3999939  
 345.79998779 340.07998657 335.91998291 339.03997803 343.19998169  
 342.15997314 341.11999512 339.55999756 345.27999878 344.23999023  
 343.72000122 342.67999268 346.31997681 337.47998047 336.44000244  
 333.83999634 334.87997437 340.59997559 359.31997681 379.59997559  
 376.47998047 374.3999939 373.87997437 375.43997192 380.11999512  
 371.27999878 379.07998657 377.51998901 375.95999146 374.91998291  
 370.23999023 368.15997314 373.35998535 378.03997803 372.31997681  
 367.63998413 341.625 343.25 322.375 337.375  
 340.125 340.25 340.375 337.875 330.375  
 335.625 337.75 395. 389.125 388.875  
 393.125 393.25 387.25 393.625 383.75  
 388.625 387.75 378.25 378.875 376.25  
 378.75 376.75 377.875 348.625 344.875  
 346.5 383.25 351.875 347.625 342.375  
 342.125 350.625 343.75 339.25 339.875  
 337.625 364.375 373.125 273.5 220.5  
 187.5 332.25 341.125 300.125 334.125  
 299.375 342.625 342.25 341.875 341.25  
 340.875 340.75 343.125 390.625 387.125  
 391.25 389.875 391.125 389.25 375.75  
 185.5 337.25 339.125 341.375 331.75  
 333.625 347.25 277.5 276. 275.5  
 321.75 310.125 295.375 328.75 384.625  
 391.75 381.125 384.125 343.875 334.375  
 329.375 340.625 343.5 333.375 342.875  
 329.875 335.125 346.25 334.625 324.75  
 339.75 311.75 342.75 332.75 335.75  
 336.875 331.875 371.75 344.75 336.125  
 372.625 329.625 321.25 331.25 332.375  
 337.125 320.75 322.875 320.125 324.25  
 333.125 334.25 343.375 334.75 328.875  
 346.375 344.25 390.875 181.5 272.  
 221. 328.125 345.75 332.125 333.75  
 322.625 338.875 312.875 176.5 271.  
 266. 266.5 270. 272.5 275.  
 179.5 178. 175.5 391.875 390.25  
 394.25 373.875 372.75 380.63998413 392.07998657  
 386.35998535 387.91998291 383.75997925 385.83999634 384.79998779  
 382.7199707 378.55999756 381.67999268 328.63998413 321.35998535  
 324.47998047 321.87997437 333.31997681 327.59997559 327.07998657  
 326.03997803 325.51998901 323.95999146 322.3999939 317.19998169  
 319.79998779 329.67999268 330.19998169 336.95999146 338.51998901  
 330.72000122 334.875 326.75 330.875 219.5  
 221.5 338.625 322.75 326.625 336.375  
 336.625 314.75 316.25 327.75 307.75  
 271.5 375.375 394.75 389.75 390.75  
 223. 219. 222.5 218. 223.5  
 220. 222. 224. 224.5 335.3999939  
 334.35998535 332.79998779 384.27999878 381.15997314 333.875  
 325.625 335.25 331.125 335.375 315.625  
 322.125 332.875 329.75 330.625 328.625  
 309.3999939 331.23999023 323.625 325.875 326.125  
 325.375 331.625 305.625 302.25 313.75  
 308.875 302.125 336.75 329.125 328.25  
 321.125 268.5 393.375 391.375 394.5  
 392.625 389.375 269. 373.375 336.25  
 338.125 182. 335.875 392.875 392.25  
 182.5 180. 177.5 269.5 365.625  
 309.75 319.625 318.625 324.875 317.75  
 317.125 327.25 333.25 319.375 327.125  
 310.25 312.625 317.625 327.375 331.375  
 321.875 316.375 267.5 263.5 396.5  
 397.25 394.125 170.5 178.5 177.  
 179. 324.125 323.75 323.875 330.125  
 330.25 397.375 395.125 397.625 396.125  
 391.625 318.125 307.125 306.25 330.75  
 327.875 324.625 325.75 377.25 326.375  
 383.23999023 310.625 306.875 331.75997925 317.25  
 317.375 322.25 313.375 332.625 325.125  
 326.875 309.875 329.25 327.625 324.375  
 311.125 303.75 319.125 313.125 291.875  
 312.375 304.375 310.75 299.625 312.75  
 302.75 274.125 304.75 325.25 304.125  
 307.625 316.125 301.625 318.875 319.875  
 396.25 323.25 392.75 180.5 395.5 ]  
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 17.1875 16.40625  
 22.65625 25.78125 53.125 32.8125 50. 51.5625 26.5625  
 35.9375 31.25 27.34375 -21.875 -28.90625 42.1875 30.46875  
 89.84375 32.03125 39.84375 35.15625 40.625 47.65625 -24.21875  
 38.28125 37.5 46.09375 39.0625 46.875 44.53125 42.96875  
 41.40625 43.75 -29.6875 -25. -26.5625 45.3125 -28.125  
 78.90625 52.34375 50.78125 53.90625 49.21875 48.4375 -35.15625  
 72.65625]  
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 25. 53.125 19.53125 14.0625  
 28.125 42.1875 32.8125 22.65625 35.9375 51.5625 14.84375  
 28.90625 30.46875 40.625 36.71875 32.03125 33.59375 31.25  
 17.1875 18.75 23.4375 25.78125 24.21875 -28.90625 13.28125  
 -22.65625 -23.4375 27.34375 35.15625 26.5625 20.3125 39.0625  
 37.5 39.84375 -21.875 -26.5625 -93.75 -24.21875 -25.78125  
 -30.46875 -25. 41.40625 52.34375 50.78125 47.65625 46.875  
 53.90625 42.96875 50. 38.28125 44.53125 45.3125 43.75  
 49.21875 46.09375 -27.34375]  
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 25.  
 19.53125 21.875 23.4375 22.65625 17.96875 21.09375 -14.0625  
 -11.71875 -15.625 -18.75 -17.1875 -20.3125 -16.40625 18.75  
 -19.53125 -22.65625 25.78125 26.5625 -17.96875 35.15625 24.21875  
 20.3125 -23.4375 -32.8125 -29.6875 -24.21875 -25.78125 -26.5625  
 -21.09375 -21.875 32.8125 33.59375 27.34375 28.125 34.375  
 -27.34375 -28.125 -30.46875 28.90625 38.28125]  
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 25. 23.4375 18.75 21.09375 14.0625  
 13.28125 22.65625 19.53125 15.625 -8.59375 11.71875 10.9375  
 -14.0625 -16.40625 -11.71875 -15.625 -14.84375 14.84375 -19.53125  
 12.5 -17.96875 21.875 17.1875 17.96875 16.40625 -25.  
 -25.78125 -21.09375 -17.1875 24.21875 20.3125 -18.75 -26.5625  
 -21.875 -22.65625 -30.46875 -20.3125 ]

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

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

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

'\nCreated new df\_dynamic by concatenating df\_part1 and df\_part2\n'

#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())  
  
'''

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

#df\_dynamic.head()

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

'\nChecked duplicates\n\nduplicates = df\_dynamic.duplicated()\nif duplicates.any():\n print("Duplicates found in the DataFrame:")\n print(df\_dynamic[duplicates])\n'

'''  
Checked duplicates  
'''  
  
duplicates = df\_part1.duplicated()  
if duplicates.any():  
 print("Duplicates found in the DataFrame:")  
 print(df\_part1[duplicates])

df\_static.columns

Index(['VehId', 'Vehicle Type', 'Vehicle Class',  
 'Engine Configuration & Displacement', 'Transmission', 'Drive Wheels',  
 'Generalized\_Weight'],  
 dtype='object')

#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  
It is handled in the load\_csv\_files\_from\_directory function.  
'''  
  
df\_dynamic\_sample = df\_part1

len(df\_dynamic\_sample)

5131987

'''  
  
push\_df\_to\_s3(df\_dynamic\_sample, 'Cleaned up VED Source Data/df\_dynamic\_sample.csv')  
  
'''

"\n\npush\_df\_to\_s3(df\_dynamic\_sample, 'Cleaned up VED Source Data/df\_dynamic\_sample.csv')\n\n"

push\_df\_to\_s3\_parquet(df\_dynamic\_sample, 'Cleaned up VED Source Data/df\_dynamic\_sample.parquet')

DataFrame uploaded to s3://s3aravindh973515031797/Cleaned up VED Source Data/df\_dynamic\_sample.parquet

'''  
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: 5131987 entries, 0 to 5131986  
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: 7887  
[ 1.58665119 1.93178629 1.71977381 ... 126.83646968 126.8464274  
 126.04467598]  
DayNum  
42.708322 9866  
1.477313 6358  
62.840782 5826  
23.953190 5771  
24.544544 5625  
 ...   
106.717923 101  
60.602664 101  
33.663761 101  
35.790547 101  
80.675041 101  
Name: count, Length: 7887, dtype: int64  
  
Unique values in column 'VehId':  
Number of unique values: 355  
[ 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  
 311 383 404 536 563 571 202 252 288 349 366 406 409 436 458 474 475 490  
 591 602 603 605 5 142 169 381 416 435 448 470 604 135 206 437 467 542  
 607 609 123 274 333 598 167 214 286 313 466 120 429 503 159 201 539 534  
 303 447 600 251 2 162 476 164 487 526 610 137 191 285 329 386 129 414  
 492 577 211 148 270 275 348 376 461 599 255 380 325 328 397 527 360 567  
 618 580 545 616 172 247 558 297 357 379 306 541 9]  
VehId  
560 174041  
371 82594  
564 68871  
575 66913  
484 65391  
 ...   
527 199  
461 174  
9 164  
558 113  
172 110  
Name: count, Length: 355, dtype: int64  
  
Unique values in column 'Trip':  
Number of unique values: 2070  
[ 706 707 1558 ... 2257 1502 1901]  
Trip  
956 13850  
884 12834  
658 12644  
1267 10646  
789 10491  
 ...   
60 106  
2355 103  
202 103  
2549 102  
2305 102  
Name: count, Length: 2070, dtype: int64  
  
Unique values in column 'Timestamp(ms)':  
Number of unique values: 56820  
[ 0 200 1100 ... 8377200 3016500 3027200]  
Timestamp(ms)  
0 7887  
3000 2741  
16000 2600  
3100 2300  
19000 2205  
 ...   
8367200 1  
8366200 1  
8365900 1  
8364800 1  
8363800 1  
Name: count, Length: 56820, dtype: int64  
  
Unique values in column 'Latitude[deg]':  
Number of unique values: 276663  
[42.27755833 42.27825528 42.2790125 ... 42.26687 42.26242389  
 42.259955 ]  
Latitude[deg]  
42.266517 1165  
42.266613 1104  
42.275244 982  
42.265952 941  
42.286994 877  
 ...   
42.286131 1  
42.240483 1  
42.247699 1  
42.300995 1  
42.241752 1  
Name: count, Length: 276663, dtype: int64  
  
Unique values in column 'Longitude[deg]':  
Number of unique values: 351148  
[-83.69874972 -83.69880306 -83.69890111 ... -83.71665333 -83.71606639  
 -83.7147225 ]  
Longitude[deg]  
-83.747713 1164  
-83.747760 1095  
-83.744237 1001  
-83.751339 920  
-83.726332 865  
 ...   
-83.769010 1  
-83.777571 1  
-83.681458 1  
-83.790251 1  
-83.714142 1  
Name: count, Length: 351148, dtype: int64  
  
Unique values in column 'Vehicle Speed[km/h]':  
Number of unique values: 17921  
[ 40. 45. 47. ... 120.15625 115.28125 85.984375]  
Vehicle Speed[km/h]  
0.000000 646098  
54.000000 80523  
56.000000 78962  
57.000000 78959  
60.000000 77198  
 ...   
127.984375 1  
127.593750 1  
124.593750 1  
110.259995 1  
112.009995 1  
Name: count, Length: 17921, dtype: int64  
  
Unique values in column 'MAF[g/sec]':  
Number of unique values: 8573  
[22.12999916 6.1500001 21.44000053 ... 72.54000092 73.88999939  
 74.79000092]  
MAF[g/sec]  
0.710000 161436  
0.870000 78271  
0.170000 74681  
0.180000 51256  
0.180000 42796  
 ...   
67.639999 1  
65.070000 1  
103.849998 1  
78.989998 1  
37.059998 1  
Name: count, Length: 8573, dtype: int64  
  
Unique values in column 'Engine RPM[RPM]':  
Number of unique values: 4272  
[2285. 2744. 1982. ... 5357. 4427. 4429.]  
Engine RPM[RPM]  
0.0 733791  
1280.0 25063  
1312.0 19957  
1184.0 18773  
1248.0 16728  
 ...   
3754.0 2  
3806.0 1  
3688.0 1  
4833.0 1  
4306.0 1  
Name: count, Length: 4272, dtype: int64  
  
Unique values in column 'Absolute Load[%]':  
Number of unique values: 423  
[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  
 1.33333328e+02 1.22352943e+02 1.66274506e+02 1.16470589e+02  
 1.34117645e+02 1.43529419e+02 1.17254906e+02 1.38823532e+02  
 1.04313728e+02 1.96078432e+00 1.28576475e+04 1.72156860e+02  
 1.08627449e+02 1.28235291e+02 1.10588234e+02 1.17647064e+00  
 1.15686279e+02 1.05490196e+02 1.07450981e+02 1.13725494e+02  
 1.10980392e+02 1.10196083e+02 2.41568634e+02 1.34509811e+02  
 1.32941177e+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 1.16862747e+02 1.14901962e+02  
 1.39215683e+02 1.41568634e+02 1.32156860e+02 1.55686279e+02  
 1.58039215e+02 1.30588242e+02 2.30588242e+02 1.73435293e+04  
 1.86274506e+02 1.52549026e+02 1.18431374e+02 1.75686279e+02  
 1.65639219e+04 1.84313736e+02 1.74117645e+02 1.63529419e+02  
 1.59215683e+02 1.48627457e+02 1.57647064e+02 1.20392159e+02  
 1.52156860e+02 1.74509811e+02 1.41960785e+02 1.29803925e+02  
 1.33725494e+02 1.44313721e+02 1.49019608e+02 1.60392166e+02  
 1.18039215e+02 1.20784317e+02 1.46274506e+02 1.23529411e+02  
 1.69803925e+02 1.56862748e+00 2.06274506e+02 1.85882355e+02  
 1.65098038e+02 1.30980392e+02 1.36141182e+04 1.40000000e+02  
 1.70980392e+02 1.60345098e+04 1.13333336e+02 1.80784317e+02  
 1.38431381e+02 1.70588242e+02 1.74901962e+02 1.65882355e+02  
 2.16470596e+02 1.61176468e+02 1.54509811e+02 1.40392166e+02  
 1.50196075e+02 1.21960785e+02 1.51372559e+02 1.46666672e+02  
 1.38039215e+02 1.47843140e+02 1.49803925e+02 1.45882355e+02  
 1.32549026e+02 1.52941177e+02 2.03529419e+02 1.59607849e+02  
 1.43137253e+02 1.48235291e+02 1.73333344e+02 1.90196075e+02]  
Absolute Load[%]  
0.000000 314936  
14.901961 61559  
15.686275 57689  
15.294118 55902  
14.509804 55764  
 ...   
141.568634 4  
146.274506 4  
543.921570 3  
165.882355 2  
16034.509766 2  
Name: count, Length: 423, dtype: int64  
  
Unique values in column 'OAT[DegC]':  
Number of unique values: 176  
[ 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. -5.5 -5. -6. -7. -1. 0. -10. -3. -4.  
 -2. 1. -3.5 -2.5 -3.25 -3.75 -4.75 -5.25 -4.5 -5.75  
 -6.25 -6.5 -6.75 -0.75 -1.75 -1.5 -2.25 -1.25 -2.75 -8.  
 0.5 -0.5 -7.5 -4.25 -9. -11. -8.25 -8.5 -9.75 -9.25  
 -12. -13. -7.75 -7.25 -10.25 -10.5 0.25 1.25 0.75 21.  
 -9.5 -11.75 -10.75 -14. -17. -16. -12.5 -12.75 -12.25 -13.5  
 -0.25 -11.5 -18. -11.25 -17.5 -17.75 -17.25 -18.5 -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 -8.75 -20.5  
 -21. 33. -40. -39. 17.5 18.25 18.5 18.75 15.75 16.25  
 16.75 14.25 17.75 -38. -37. -36. 34. ]  
OAT[DegC]  
 0.00 97489  
 2.00 89414  
 4.00 89027  
 1.00 87395  
 6.00 87274  
 ...   
 17.50 50  
-36.00 35  
 18.75 13  
-21.25 10  
-19.75 2  
Name: count, Length: 176, dtype: int64  
  
Unique values in column 'Fuel Rate[L/hr]':  
Number of unique values: 1570  
[ nan 0. 6.69228172 ... 4.62211037 3.82075405 1.49300373]  
Fuel Rate[L/hr]  
0.000000 135683  
1.640873 219  
1.645643 182  
1.669493 172  
1.745813 110  
 ...   
4.049713 2  
3.181577 2  
9.640129 1  
10.408096 1  
11.228533 1  
Name: count, Length: 1570, dtype: int64  
  
Unique values in column 'Air Conditioning Power[kW]':  
Number of unique values: 47  
[ 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 6.27999973 6.35999966 6.11999989 6. 5.96000004  
 5.87999964 5.51999998 5.35999966 5.44000006 5.4000001 5.27999973  
 5.31999969 0.12 4.75999975 4.19999981 4.83999968 5.19999981]  
Air Conditioning Power[kW]  
0.00 117690  
0.20 8076  
0.24 5352  
0.16 3378  
0.28 3071  
6.20 2528  
5.88 2485  
5.92 2280  
5.76 2111  
6.24 1641  
6.16 1575  
0.36 1555  
5.84 1519  
5.80 1474  
5.96 1425  
6.08 1239  
5.72 1202  
0.32 1196  
0.40 859  
6.04 844  
5.68 841  
6.12 806  
5.36 776  
6.00 762  
5.28 515  
0.04 493  
5.40 490  
0.12 453  
5.44 403  
5.52 369  
5.56 306  
5.60 304  
0.44 300  
5.32 220  
5.64 206  
5.20 116  
4.84 113  
4.20 104  
0.60 96  
6.28 91  
4.76 89  
6.32 88  
0.48 82  
0.08 72  
6.36 51  
0.56 20  
0.52 7  
Name: count, dtype: int64  
  
Unique values in column 'Air Conditioning Power[Watts]':  
Number of unique values: 52  
[ 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. 600. 1200. 1100. 1150.  
 1400. 50. 1800. 80. 1300. 1950. 1900. 1850. 1250. 2500. 2050. 2350.  
 1700. 1750. 520. 40. 1500.]  
Air Conditioning Power[Watts]  
0.0 453528  
400.0 11170  
200.0 10013  
150.0 9219  
450.0 6289  
350.0 5600  
100.0 4818  
800.0 4016  
1050.0 3133  
1100.0 3069  
1000.0 3004  
900.0 2827  
750.0 2782  
160.0 2708  
850.0 2702  
240.0 2343  
120.0 2126  
500.0 2013  
650.0 1537  
950.0 1506  
700.0 1342  
320.0 1310  
300.0 1267  
280.0 1101  
550.0 907  
1850.0 860  
1800.0 818  
80.0 810  
1200.0 741  
600.0 700  
1250.0 671  
250.0 666  
2050.0 527  
1150.0 485  
1700.0 464  
1900.0 372  
1300.0 364  
50.0 317  
1950.0 275  
1350.0 259  
360.0 245  
2500.0 223  
1750.0 166  
480.0 153  
1400.0 107  
1500.0 105  
40.0 90  
440.0 79  
640.0 69  
2350.0 55  
520.0 48  
560.0 22  
Name: count, dtype: int64  
  
Unique values in column 'Heater Power[Watts]':  
Number of unique values: 17  
[ nan 2250. 2000. 1750. 1500. 1250. 1000. 750. 500. 250. 0. 4000.  
 3500. 3750. 3000. 3250. 2750. 2500.]  
Heater Power[Watts]  
0.0 34351  
250.0 10765  
500.0 7654  
750.0 6216  
1000.0 4186  
1500.0 2816  
1250.0 2754  
1750.0 2666  
3000.0 2417  
3500.0 1275  
3250.0 1140  
2250.0 850  
2750.0 714  
2000.0 636  
2500.0 510  
3750.0 91  
4000.0 86  
Name: count, dtype: int64  
  
Unique values in column 'HV Battery Current[A]':  
Number of unique values: 14977  
[ nan -21.5 23.5 ... -165.30000305 -155.55000305  
 -174.30000305]  
HV Battery Current[A]  
-1.500000 4064  
-3.000000 2502  
 0.000000 2114  
-4.500000 1968  
-6.000000 1878  
 ...   
-48.440002 1  
 37.019989 1  
-54.380005 1  
-35.809998 1  
 76.610001 1  
Name: count, Length: 14977, dtype: int64  
  
Unique values in column 'HV Battery SOC[%]':  
Number of unique values: 3555  
[ nan 96.34146881 95.97561646 ... 63.2911377 56.68354034  
 44.93670654]  
HV Battery SOC[%]  
0.000000 179192  
1.000000 26923  
2.000000 26324  
3.000000 16320  
0.392157 10209  
 ...   
67.924042 1  
34.627850 1  
64.513931 1  
47.382282 1  
31.974682 1  
Name: count, Length: 3555, dtype: int64  
  
Unique values in column 'HV Battery Voltage[V]':  
Number of unique values: 974  
[ nan 386. 390.5 384.5 387.  
 382.5 389.5 387.5 378.5 383.5  
 389. 388.5 381.5 291.5 300.  
 294.5 292.5 287.5 296.5 293.  
 296. 289. 301.5 295. 295.5  
 284. 293.5 297. 279. 290.  
 292. 304. 294. 298.5 297.5  
 291. 307.5 285. 300.5 288.  
 309.5 299. 299.5 301. 302.  
 274. 306. 302.5 309. 298.  
 308. 287. 286. 286.5 290.5  
 282.5 283. 303.5 288.5 345.  
 348. 346. 347. 343. 352.  
 359. 350. 351. 353. 356.  
 338. 349. 360. 339. 341.  
 337. 342. 344. 330. 332.  
 333. 340. 329. 331. 326.  
 334. 336. 357. 335. 358.  
 359.5 361.25 361.75 357.375 360.5  
 368.125 362.25 356.375 358.875 359.25  
 371.875 356.875 354.75 361.375 361.625  
 352.25 354.625 366.125 352.375 357.875  
 357.5 358.25 360.75 361. 371.25  
 362. 361.5 357.625 367.5 360.375  
 357.75 360.25 361.125 359.875 365.25  
 361.875 354.125 355.875 365.75 360.875  
 360.625 352.75 354. 354.5 363.75  
 359.625 357.125 355.125 359.375 358.75  
 358.5 362.625 358.125 355.25 362.875  
 354.375 355. 357.25 352.875 354.875  
 359.125 350.25 358.625 358.375 356.25  
 355.75 356.75 356.125 355.5 356.625  
 355.625 356.5 352.125 353.875 362.75  
 353.75 351.5 352.625 351.375 349.5  
 363.375 348.5 338.75 339.625 332.5  
 341.75 344.625 347.75 328.375 348.125  
 345.25 348.75 338.375 338.25 347.125  
 339.375 346.75 366.375 369.25 374.75  
 375.125 370.5 370.625 367. 359.75  
 362.5 365.875 366.875 367.375 368.375  
 363. 364.25 369. 378.375 366.625  
 367.25 367.625 368. 367.875 368.625  
 365. 353.25 364.625 366.5 367.125  
 354.25 353.625 347.375 362.125 362.375  
 349.375 364.5 353.125 351.75 350.875  
 380. 371.5 382. 379.5 377.5  
 376.5 375.5 371. 374.5 379.  
 373.5 374. 373. 385. 378.  
 381. 375. 377. 384. 380.5  
 383. 331.5 322.5 326.5 316.  
 324.5 320. 324. 327.5 328.5  
 325. 322. 323.5 321.5 325.5  
 316.5 318.5 314.5 315.5 310.  
 318. 321. 323. 313. 319.5  
 334.5 310.5 307. 317.5 311.  
 305.5 319. 315. 313.5 314.  
 312. 308.5 305. 311.5 304.5  
 317. 200. 195. 198.5 201.5  
 199.5 207. 208.5 209.5 215.  
 198. 197. 209. 213.5 208.  
 196.5 205.5 201. 200.5 199.  
 206.5 207.5 217.5 195.5 206.  
 218.5 193.5 191.5 289.5 284.5  
 303. 285.5 306.5 312.5 283.5  
 278. 280.5 383.625 385.375 388.375  
 392.375 386.75 384.25 372. 368.75  
 373.625 377.125 381.875 395.375 376.125  
 377.75 376.875 378.625 382.25 385.75  
 372.125 380.875 384.75 388. 388.75  
 383.875 379.125 192. 202. 216.5  
 194.5 190. 194. 197.5 193.  
 188. 196. 202.5 210.5 203.5  
 211. 187. 212. 186. 190.5  
 191. 204.5 192.5 210. 204.  
 188.5 189.5 181. 184. 205.  
 186.5 214.5 203. 211.5 212.5  
 189. 338.5 337.5 333.5 336.5  
 330.5 327. 335.5 342.5 341.5  
 345.5 329.5 328. 183. 215.5  
 213. 216. 217. 339.5 320.5  
 371.375 380.25 369.75 363.5 364.125  
 370.125 366.25 371.625 371.125 364.75  
 370.375 370.25 370.875 366. 363.125  
 363.625 363.25 360.125 185. 184.5  
 183.5 376. 370. 385.5 372.5  
 368.5 369.5 365.5 364. 282.  
 281.5 281. 278.5 277. 270.5  
 276.5 274.5 385.875 393.75 386.125  
 380.75 381.25 382.375 385.25 386.625  
 386.875 387.625 382.75 381.75 384.875  
 383.375 386.25 388.25 385.125 390.125  
 387.875 386.375 386.5 384.375 379.25  
 383.125 382.125 381.625 376.375 379.875  
 380.125 379.375 380.625 379.75 380.375  
 379.625 374.875 373.75 374.625 377.375  
 375.25 372.25 369.625 367.75 370.75  
 387.375 376.625 378.125 368.875 374.25  
 279.5 280. 340.5 273. 352.5  
 353.5 350.75 351.25 349.875 355.375  
 350.125 349.625 349.125 349.75 350.5  
 344.5 346.875 348.25 347.875 348.875  
 351.125 345.125 346.625 348.375 344.125  
 345.375 345.625 346.125 345.875 349.25  
 351.625 393. 392.5 391. 391.5  
 390. 392. 393.5 394. 347.5  
 362.95999146 350.47998047 351.51998901 360.87997437 360.35998535  
 357.23999023 356.72000122 372.83999634 369.19998169 371.79998779  
 370.75997925 367.11999512 362.43997192 366.07998657 366.59997559  
 365.55999756 365.03997803 364.51998901 361.91998291 355.15997314  
 353.07998657 354.63998413 349.44000244 354.11999512 357.75997925  
 368.67999268 352.55999756 363.47998047 358.79998779 355.67999268  
 356.19998169 369.72000122 359.83999634 385.625 382.875  
 375.625 381.375 389.625 388.125 374.375  
 382.625 390.375 394.625 377.625 214.  
 375.875 374.125 363.875 344.375 353.375  
 350.375 343.625 369.125 368.25 369.875  
 372.875 372.375 365.375 366.75 369.375  
 373.25 365.125 364.875 361.3999939 358.27999878  
 347.35998535 349.95999146 347.87997437 353.59997559 341.63998413  
 352.03997803 344.75997925 346.83999634 348.91998291 348.3999939  
 345.79998779 340.07998657 335.91998291 339.03997803 343.19998169  
 342.15997314 341.11999512 339.55999756 345.27999878 344.23999023  
 343.72000122 342.67999268 346.31997681 337.47998047 336.44000244  
 333.83999634 334.87997437 340.59997559 359.31997681 379.59997559  
 376.47998047 374.3999939 373.87997437 375.43997192 380.11999512  
 371.27999878 379.07998657 377.51998901 375.95999146 374.91998291  
 370.23999023 368.15997314 373.35998535 378.03997803 372.31997681  
 367.63998413 341.625 343.25 322.375 337.375  
 340.125 340.25 340.375 337.875 330.375  
 335.625 337.75 395. 389.125 388.875  
 393.125 393.25 387.25 393.625 383.75  
 388.625 387.75 378.25 378.875 376.25  
 378.75 376.75 377.875 348.625 344.875  
 346.5 383.25 351.875 347.625 342.375  
 342.125 350.625 343.75 339.25 339.875  
 337.625 364.375 373.125 273.5 220.5  
 187.5 332.25 341.125 300.125 334.125  
 299.375 342.625 342.25 341.875 341.25  
 340.875 340.75 343.125 390.625 387.125  
 391.25 389.875 391.125 389.25 375.75  
 185.5 337.25 339.125 341.375 331.75  
 333.625 347.25 277.5 276. 275.5  
 321.75 310.125 295.375 328.75 384.625  
 391.75 381.125 384.125 343.875 334.375  
 329.375 340.625 343.5 333.375 342.875  
 329.875 335.125 346.25 334.625 324.75  
 339.75 311.75 342.75 332.75 335.75  
 336.875 331.875 371.75 344.75 336.125  
 372.625 329.625 321.25 331.25 332.375  
 337.125 320.75 322.875 320.125 324.25  
 333.125 334.25 343.375 334.75 328.875  
 346.375 344.25 390.875 181.5 272.  
 221. 328.125 345.75 332.125 333.75  
 322.625 338.875 312.875 176.5 271.  
 266. 266.5 270. 272.5 275.  
 179.5 178. 175.5 391.875 390.25  
 394.25 373.875 372.75 380.63998413 392.07998657  
 386.35998535 387.91998291 383.75997925 385.83999634 384.79998779  
 382.7199707 378.55999756 381.67999268 328.63998413 321.35998535  
 324.47998047 321.87997437 333.31997681 327.59997559 327.07998657  
 326.03997803 325.51998901 323.95999146 322.3999939 317.19998169  
 319.79998779 329.67999268 330.19998169 336.95999146 338.51998901  
 330.72000122 334.875 326.75 330.875 219.5  
 221.5 338.625 322.75 326.625 336.375  
 336.625 314.75 316.25 327.75 307.75  
 271.5 375.375 394.75 389.75 390.75  
 223. 219. 222.5 218. 223.5  
 220. 222. 224. 224.5 335.3999939  
 334.35998535 332.79998779 384.27999878 381.15997314 333.875  
 325.625 335.25 331.125 335.375 315.625  
 322.125 332.875 329.75 330.625 328.625  
 309.3999939 331.23999023 323.625 325.875 326.125  
 325.375 331.625 305.625 302.25 313.75  
 308.875 302.125 336.75 329.125 328.25  
 321.125 268.5 393.375 391.375 394.5  
 392.625 389.375 269. 373.375 336.25  
 338.125 182. 335.875 392.875 392.25  
 182.5 180. 177.5 269.5 365.625  
 309.75 319.625 318.625 324.875 317.75  
 317.125 327.25 333.25 319.375 327.125  
 310.25 312.625 317.625 327.375 331.375  
 321.875 316.375 267.5 263.5 396.5  
 397.25 394.125 170.5 178.5 177.  
 179. 324.125 323.75 323.875 330.125  
 330.25 397.375 395.125 397.625 396.125  
 391.625 318.125 307.125 306.25 330.75  
 327.875 324.625 325.75 377.25 326.375  
 383.23999023 310.625 306.875 331.75997925 317.25  
 317.375 322.25 313.375 332.625 325.125  
 326.875 309.875 329.25 327.625 324.375  
 311.125 303.75 319.125 313.125 291.875  
 312.375 304.375 310.75 299.625 312.75  
 302.75 274.125 304.75 325.25 304.125  
 307.625 316.125 301.625 318.875 319.875  
 396.25 323.25 392.75 180.5 395.5 ]  
HV Battery Voltage[V]  
197.000000 23204  
197.500000 15485  
198.000000 13250  
195.500000 10665  
198.500000 9714  
 ...   
299.625000 6  
392.079987 6  
396.125000 6  
383.239990 6  
318.125000 3  
Name: count, Length: 974, dtype: int64  
  
Unique values in column 'Short Term Fuel Trim Bank 1[%]':  
Number of unique values: 112  
[ 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 17.1875 16.40625  
 22.65625 25.78125 53.125 32.8125 50. 51.5625 26.5625  
 35.9375 31.25 27.34375 -21.875 -28.90625 42.1875 30.46875  
 89.84375 32.03125 39.84375 35.15625 40.625 47.65625 -24.21875  
 38.28125 37.5 46.09375 39.0625 46.875 44.53125 42.96875  
 41.40625 43.75 -29.6875 -25. -26.5625 45.3125 -28.125  
 78.90625 52.34375 50.78125 53.90625 49.21875 48.4375 -35.15625  
 72.65625]  
Short Term Fuel Trim Bank 1[%]  
 0.00000 1009728  
 0.78125 288772  
-0.78125 279549  
-1.56250 254467  
 1.56250 245243  
 ...   
 89.84375 7  
-35.15625 7  
 53.90625 6  
 48.43750 6  
 72.65625 5  
Name: count, Length: 112, dtype: int64  
  
Unique values in column 'Short Term Fuel Trim Bank 2[%]':  
Number of unique values: 107  
[ 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 25. 53.125 19.53125 14.0625  
 28.125 42.1875 32.8125 22.65625 35.9375 51.5625 14.84375  
 28.90625 30.46875 40.625 36.71875 32.03125 33.59375 31.25  
 17.1875 18.75 23.4375 25.78125 24.21875 -28.90625 13.28125  
 -22.65625 -23.4375 27.34375 35.15625 26.5625 20.3125 39.0625  
 37.5 39.84375 -21.875 -26.5625 -93.75 -24.21875 -25.78125  
 -30.46875 -25. 41.40625 52.34375 50.78125 47.65625 46.875  
 53.90625 42.96875 50. 38.28125 44.53125 45.3125 43.75  
 49.21875 46.09375 -27.34375]  
Short Term Fuel Trim Bank 2[%]  
 0.00000 191510  
 0.78125 88056  
-0.78125 85482  
-1.56250 82430  
 1.56250 77372  
 ...   
-25.78125 9  
-30.46875 7  
-26.56250 7  
 46.09375 6  
-27.34375 6  
Name: count, Length: 107, dtype: int64  
  
Unique values in column 'Long Term Fuel Trim Bank 1[%]':  
Number of unique values: 81  
[ 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 25.  
 19.53125 21.875 23.4375 22.65625 17.96875 21.09375 -14.0625  
 -11.71875 -15.625 -18.75 -17.1875 -20.3125 -16.40625 18.75  
 -19.53125 -22.65625 25.78125 26.5625 -17.96875 35.15625 24.21875  
 20.3125 -23.4375 -32.8125 -29.6875 -24.21875 -25.78125 -26.5625  
 -21.09375 -21.875 32.8125 33.59375 27.34375 28.125 34.375  
 -27.34375 -28.125 -30.46875 28.90625 38.28125]  
Long Term Fuel Trim Bank 1[%]  
 0.00000 312845  
-1.56250 290648  
 1.56250 280048  
 0.78125 277934  
-0.78125 274908  
 ...   
 27.34375 38  
 32.81250 35  
 34.37500 23  
-25.78125 15  
-29.68750 9  
Name: count, Length: 81, dtype: int64  
  
Unique values in column 'Long Term Fuel Trim Bank 2[%]':  
Number of unique values: 66  
[ 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 25. 23.4375 18.75 21.09375 14.0625  
 13.28125 22.65625 19.53125 15.625 -8.59375 11.71875 10.9375  
 -14.0625 -16.40625 -11.71875 -15.625 -14.84375 14.84375 -19.53125  
 12.5 -17.96875 21.875 17.1875 17.96875 16.40625 -25.  
 -25.78125 -21.09375 -17.1875 24.21875 20.3125 -18.75 -26.5625  
 -21.875 -22.65625 -30.46875 -20.3125 ]  
Long Term Fuel Trim Bank 2[%]  
 0.00000 97716  
-0.78125 73123  
 1.56250 72617  
 0.78125 72542  
 2.34375 71083  
 ...   
-18.75000 47  
-25.78125 45  
-20.31250 25  
-21.87500 13  
-26.56250 11  
Name: count, Length: 66, dtype: int64  
  
Unique values in column 'Vehicle Type':  
Number of unique values: 4  
['ICE' 'EV' 'PHEV' 'HEV']  
Vehicle Type  
ICE 3253818  
HEV 1158475  
PHEV 640567  
EV 79127  
Name: count, dtype: int64  
  
Unique values in column 'Vehicle Class':  
Number of unique values: 1  
['Car' nan]  
Vehicle Class  
Car 764281  
Name: count, dtype: int64  
  
Unique values in column 'Engine Configuration & Displacement':  
Number of unique values: 69  
['4-FI 1.5L' 'ELECTRIC' '4-GAS/ELECTRIC 2.0L' '4-GAS/ELECTRIC 1.8L'  
 '4-GAS/ELECTRIC 2.4L' '4-FI 2.4L' '4-FI 2.5L' '5-FI 2.5L' '6-FI 3.5L'  
 '4-FI 2.2L' '4-FI 2.3L T/C' '8-FI 5.4L' '4-FI 2.0L' '6-FI 3.3L'  
 '6-FI 3.8L' '4-GAS/ELECTRIC 2.3L' '4-FI T/C 1.4L' '6-FI 3.0L' '6-FI 3.6L'  
 '6-FI 3.7L' '6-FI 2.7L' '4-GAS/ELECTRIC 1.5L' '4-FI 2.3L ULEV'  
 '4-FI 1.6L T/C' '8-FI 5.3L' '4-GAS/ELECTRIC 2.5L' '4-FI 1.4L T/C'  
 '6-GAS/ELECTRIC 3.5L' '4-FI 1.8L' '4-FI T/C 2.0L' '4-FI S/C 1.8L GAS'  
 '4-FI 1.6L' '4-GAS/ELECTRIC 1.4L' 'I4 2.4L Flex Fuel' '8-4V/FI 6.0L'  
 '10-FI 6.8L' '6-FI 3.4L' '4-FI 1.3L GAS/ELEC.' '8-FI 4.6L'  
 '4-FI 2.0L T/C' '8-FI 4.8L' '4-FI 2.0L PZEV' '8-FI 5.7L HEMI'  
 '2.3L Gasoline I4' '6-FI 4.3L' '6-EFI 4.2L ' '8-EFI 5.0L' '6-EFI 3.0L'  
 '6-GAS/ELECTRIC 3.3L' '2.4L' '3.0L 6cyl 4A' 'H-4 2.0 L/122' '8-FI 4.7L'  
 '6-242-MFI 4.0L' '4-FI 1.5L T/C' '5-FI 2.5L PZEV'  
 '8-FI 5.7L HEMI (Hemi engine)' '4-FI 2.3L' '6-FI 3.1L' '6-FI 4.2L'  
 '3-FI 1.0L GAS/ELEC.' 'V6 4.0L' 'V6 3.1L' 'V8 4.7L' 'V6 3.0L' 'V6 3.8L'  
 'I4 2.2L' 'V6 3.5L' '4-GAS/ELECTRIC 1.6L']  
Engine Configuration & Displacement  
4-GAS/ELECTRIC 1.8L 616671  
4-FI 2.4L 501575  
6-FI 3.5L 474698  
4-FI 2.5L 423137  
4-GAS/ELECTRIC 1.5L 402093  
 ...   
4-FI 2.3L 2504  
4-FI 1.5L T/C 1932  
8-EFI 5.0L 1473  
8-FI 4.7L 873  
4-GAS/ELECTRIC 1.6L 113  
Name: count, Length: 69, dtype: int64  
  
Unique values in column 'Transmission':  
Number of unique values: 16  
['5-SP MANUAL' nan 'CVT' '5-SP AUTOMATIC' '6-SP ECT AUTOMATIC' 'AUTOMATIC'  
 '5-SP ECT AUTOMATIC' 'FULL TIME 4WD AUTOMATIC' 'AUTOMATIC/CVT'  
 '4-SP AUTOMATIC' '9-SP Automatic' '6-SP AUTOMATIC' 'FULL TIME 4WD MANUAL'  
 '4-SP Automatic' '5-SP Automatic' '5-SP AWD MANUAL' '6-SP AWD MANUAL']  
Transmission  
CVT 658224  
5-SP AUTOMATIC 313895  
AUTOMATIC/CVT 224648  
4-SP AUTOMATIC 71172  
4-SP Automatic 48554  
5-SP MANUAL 42376  
AUTOMATIC 41536  
6-SP AUTOMATIC 32636  
6-SP ECT AUTOMATIC 31286  
5-SP ECT AUTOMATIC 27380  
FULL TIME 4WD MANUAL 17899  
FULL TIME 4WD AUTOMATIC 13915  
9-SP Automatic 9575  
5-SP Automatic 8009  
5-SP AWD MANUAL 7456  
6-SP AWD MANUAL 2860  
Name: count, dtype: int64  
  
Unique values in column 'Drive Wheels':  
Number of unique values: 1  
[nan 'FWD']  
Drive Wheels  
FWD 719694  
Name: count, dtype: int64  
  
Unique values in column 'Generalized\_Weight':  
Number of unique values: 9  
[2500. 3500. 4000. 3000. 4500. nan 5000. 6000. 5500. 2000.]  
Generalized\_Weight  
3500.0 1527308  
3000.0 1439246  
4000.0 884200  
2500.0 529405  
4500.0 495158  
5000.0 55217  
5500.0 43400  
6000.0 4846  
2000.0 4357  
Name: count, dtype: int64

df.head().to\_dict()

{'DayNum': {0: 1.58665119444,  
 1: 1.58665119444,  
 2: 1.58665119444,  
 3: 1.58665119444,  
 4: 1.58665119444},  
 'VehId': {0: 8, 1: 8, 2: 8, 3: 8, 4: 8},  
 'Trip': {0: 706, 1: 706, 2: 706, 3: 706, 4: 706},  
 'Timestamp(ms)': {0: 0, 1: 200, 2: 1100, 3: 2100, 4: 4200},  
 'Latitude[deg]': {0: 42.2775583333,  
 1: 42.2775583333,  
 2: 42.2775583333,  
 3: 42.2775583333,  
 4: 42.2775583333},  
 'Longitude[deg]': {0: -83.6987497222,  
 1: -83.6987497222,  
 2: -83.6987497222,  
 3: -83.6987497222,  
 4: -83.6987497222},  
 'Vehicle Speed[km/h]': {0: 40.0, 1: 40.0, 2: 45.0, 3: 47.0, 4: 48.0},  
 'MAF[g/sec]': {0: 22.1299991608,  
 1: 22.1299991608,  
 2: 22.1299991608,  
 3: 6.15000009537,  
 4: 21.4400005341},  
 'Engine RPM[RPM]': {0: 2285.0, 1: 2285.0, 2: 2285.0, 3: 2744.0, 4: 1982.0},  
 'Absolute Load[%]': {0: 49.0196075439,  
 1: 67.4509811401,  
 2: 67.4509811401,  
 3: 67.4509811401,  
 4: 67.4509811401},  
 'OAT[DegC]': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 '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: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'Short Term Fuel Trim Bank 2[%]': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'Long Term Fuel Trim Bank 1[%]': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'Long Term Fuel Trim Bank 2[%]': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'Vehicle Type': {0: 'ICE', 1: 'ICE', 2: 'ICE', 3: 'ICE', 4: 'ICE'},  
 'Vehicle Class': {0: 'Car', 1: 'Car', 2: 'Car', 3: 'Car', 4: 'Car'},  
 'Engine Configuration & Displacement': {0: '4-FI 1.5L',  
 1: '4-FI 1.5L',  
 2: '4-FI 1.5L',  
 3: '4-FI 1.5L',  
 4: '4-FI 1.5L'},  
 'Transmission': {0: '5-SP MANUAL',  
 1: '5-SP MANUAL',  
 2: '5-SP MANUAL',  
 3: '5-SP MANUAL',  
 4: '5-SP MANUAL'},  
 'Drive Wheels': {0: nan, 1: nan, 2: nan, 3: nan, 4: nan},  
 'Generalized\_Weight': {0: 2500.0, 1: 2500.0, 2: 2500.0, 3: 2500.0, 4: 2500.0}}

'''  
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', 'Mild', 'Warm', 'Hot', 'Cold', 'Extremely Cold'],  
 dtype=object)

df.info()

<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 5131987 entries, 0 to 5131986  
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)

'''  
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]']

df.columns

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',  
 'HV Battery Power[Watts]'],  
 dtype='object')

'''  
push\_df\_to\_s3(df, 'Cleaned up VED Source Data/df\_VED.csv')  
'''

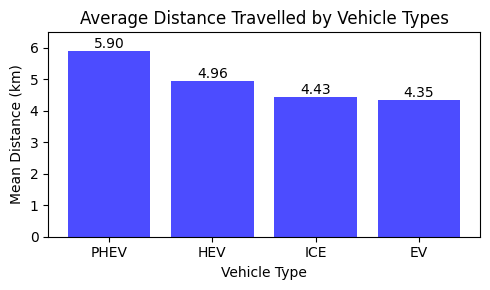
"\npush\_df\_to\_s3(df, 'Cleaned up VED Source Data/df\_VED.csv')\n"

# 4 Sample plots

'''  
Calculate and display the average distance travelled for each vehicle type.  
This is done by grouping the DataFrame by 'Vehicle Type' and computing the mean of 'Distance[km]'.  
The result is then sorted in descending order of mean distance.  
'''  
  
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 5.900412  
1 HEV 4.959184  
2 ICE 4.428779  
0 EV 4.351690

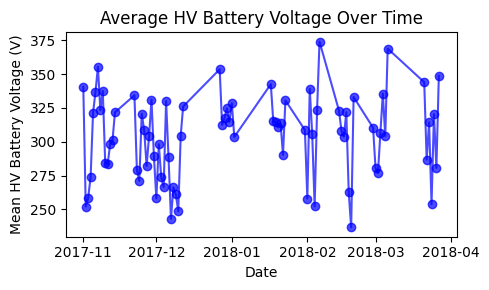
'''  
Plot a bar chart showing the average distance travelled by each vehicle type.  
- Uses matplotlib to create a bar plot of mean distance per vehicle type.  
- Sets figure size, axis labels, and title.  
- Limits y-axis to 10% above the maximum mean distance.  
- Annotates each bar with its value.  
- Displays the plot.  
'''  
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()



"""  
Calculate the average HV Battery Voltage for each day:  
- Convert the 'Date' column to datetime format.  
- Group the DataFrame by day and compute the mean of 'HV Battery Voltage[V]'.  
- Reset the index and convert the period index back to timestamps.  
- Sort the resulting DataFrame by date.  
"""  
  
df['Date'] = pd.to\_datetime(df['Date'])  
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 2017-11-01 340.664036  
1 2017-11-02 252.101365  
2 2017-11-03 258.550106  
3 2017-11-04 273.978090  
4 2017-11-05 321.147487  
.. ... ...  
73 2018-03-24 253.762327  
74 2018-03-25 320.406958  
75 2018-03-26 280.713340  
76 2018-03-27 348.769579  
77 2018-03-28 NaN  
  
[78 rows x 2 columns]

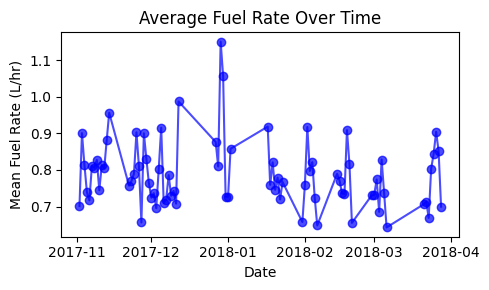
'''  
Plot the average HV Battery Voltage over time:  
- Uses matplotlib to create a line plot of mean HV Battery Voltage per day.  
- Sets figure size, axis labels, and title.  
- Displays the plot.  
'''  
  
import matplotlib.pyplot as plt  
  
plt.figure(figsize=(5, 3))  
plt.plot(df\_eot['Date'], 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()



"""  
Calculate the average Fuel Consumption Rate (FCR) for each day:  
- Ensure the 'Date' column is in datetime format.  
- Group the DataFrame by day and compute the mean of 'FCR'.  
- Reset the index and convert the period index back to timestamps.  
- Sort the resulting DataFrame by date.  
"""  
  
df['Date'] = pd.to\_datetime(df['Date'])  
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 = df\_eof[df\_eof['Date'] != df\_eof['Date'].min()] # Starting Date with FCR 0 is filtered out  
df\_eof

Date FCR  
1 2017-11-02 0.701036  
2 2017-11-03 0.900423  
3 2017-11-04 0.814091  
4 2017-11-05 0.738375  
5 2017-11-06 0.716818  
.. ... ...  
73 2018-03-24 0.803490  
74 2018-03-25 0.843573  
75 2018-03-26 0.904103  
76 2018-03-27 0.851521  
77 2018-03-28 0.699542  
  
[77 rows x 2 columns]

'''  
Plot the average Fuel Consumption Rate (FCR) over time:  
- Uses matplotlib to create a line plot of mean FCR per day.  
- Sets figure size, axis labels, and title.  
- Displays the plot.  
'''  
  
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()



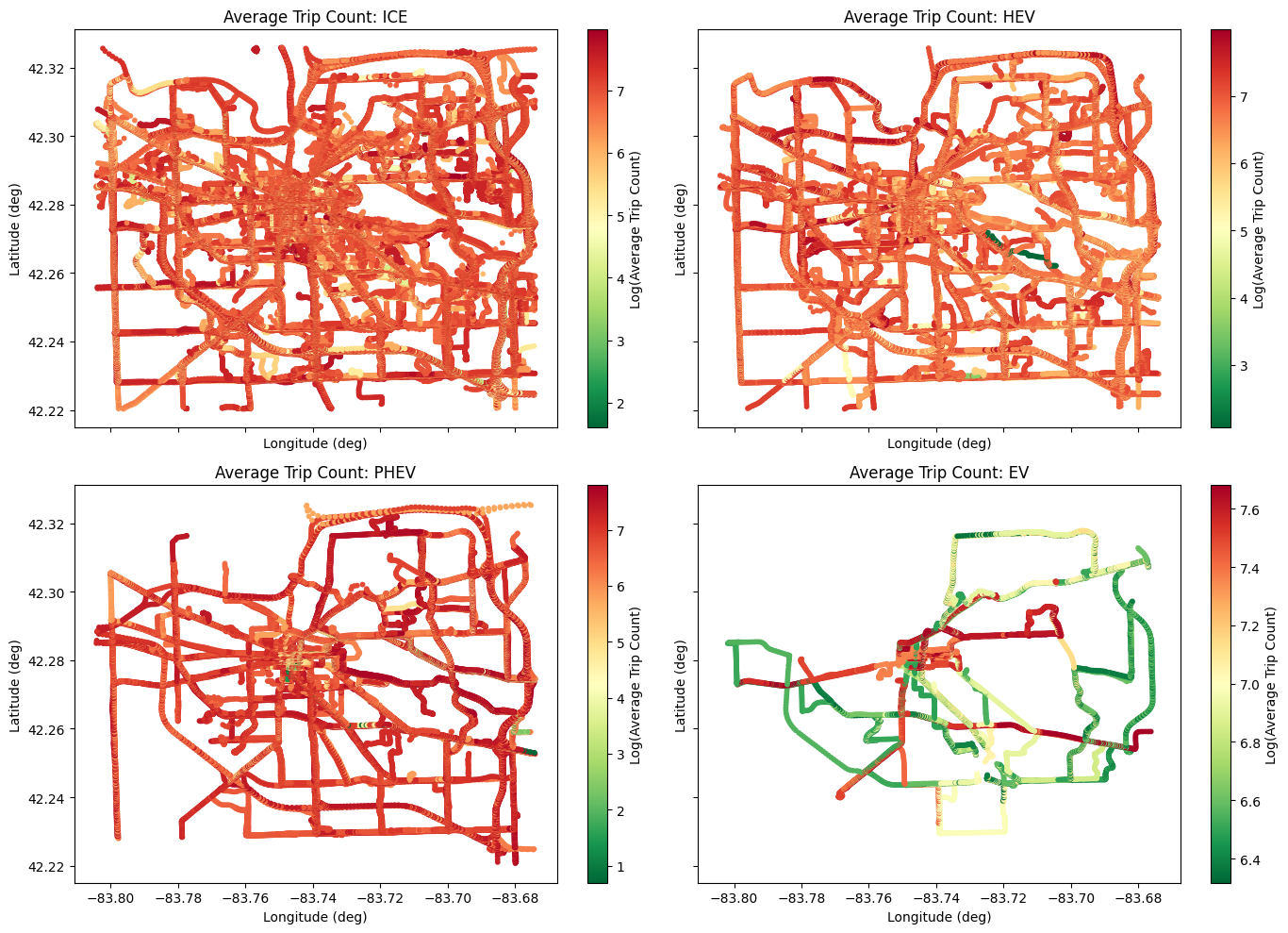
'''  
Average Trip Count by Location (Heat Map)  
'''  
  
df\_map = df.groupby(['Latitude[deg]', 'Longitude[deg]','Vehicle Type'])['Trip'].mean().reset\_index()  
df\_map

Latitude[deg] Longitude[deg] Vehicle Type Trip  
0 42.220305 -83.760323 ICE 1661.0  
1 42.220306 -83.767207 HEV 1502.0  
2 42.220316 -83.734527 ICE 1955.0  
3 42.220316 -83.760919 ICE 1399.0  
4 42.220321 -83.796176 ICE 506.0  
... ... ... ... ...  
984174 42.325675 -83.756705 ICE 1983.0  
984175 42.325766 -83.749093 ICE 1983.0  
984176 42.325775 -83.749506 ICE 1983.0  
984177 42.325780 -83.756816 ICE 1983.0  
984178 42.325796 -83.749125 ICE 1089.0  
  
[984179 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 42.220316 -83.734527 ICE 1955.0  
3 42.220316 -83.760919 ICE 1399.0  
4 42.220321 -83.796176 ICE 506.0

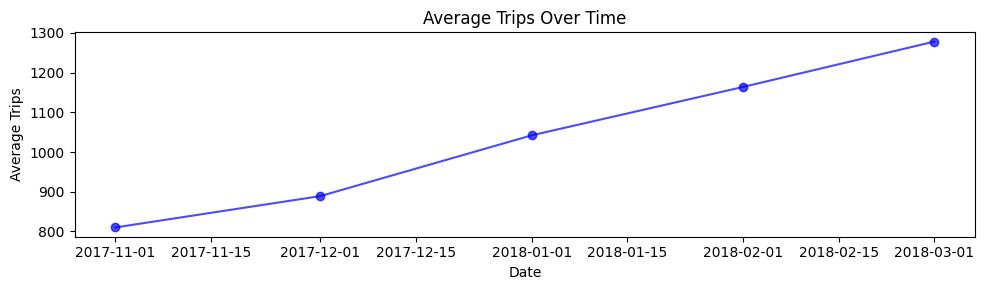
'''  
The code visualized the average trip count by location for each vehicle type using a scatter plot on subplots.   
It iterated over unique vehicle types, filtered the data for each type, and plotted longitude and latitude with color representing the logarithm of the average trip count.   
It set 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 2017-11-01 810.211164  
1 2017-12-01 889.058995  
2 2018-01-01 1042.154448  
3 2018-02-01 1164.122802  
4 2018-03-01 1278.197673

'''  
Plot the average number of trips over time using a line plot.  
Set figure size, plot the data with markers and lines, set the title and axis labels, adjust layout, and display the plot.  
'''  
import matplotlib.pyplot as plt  
  
plt.figure(figsize=(10, 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()



push\_df\_to\_s3\_parquet(df, 'Cleaned up VED Source Data/df\_VED.parquet')

DataFrame uploaded to s3://s3aravindh973515031797/Cleaned up VED Source Data/df\_VED.parquet

# 5 Exploratory Data Analysis

'''  
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 -6335.480657   
12 Mild EV -4438.888148   
0 Cold EV -6722.119330   
18 Warm PHEV -5595.309732   
11 Hot PHEV -4126.723111   
9 Extremely Cold PHEV -1956.295121   
3 Cold PHEV -3935.721770   
15 Mild PHEV -4173.067713   
7 Cool PHEV -3655.948852   
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[%] \  
4 381.597996 311.882686 79.973926   
12 36.930041 121.851852 77.668316   
0 553.080897 1153.021043 67.969506   
18 790.499485 NaN 63.454865   
11 746.000000 NaN 41.616707   
9 0.000000 NaN 32.569278   
3 4.651720 NaN 32.000160   
15 262.509246 NaN 30.322495   
7 19.399815 NaN 28.368700   
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   
9 5.772301   
3 1.625275   
15 0.728830   
7 1.012136   
1 0.545551   
2 0.878877   
5 0.554850   
6 0.844587   
8 0.813474   
10 0.645307   
13 0.524736   
14 0.826005   
16 0.193525   
17 1.165276

'''  
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] \  
2967 1565 PHEV 42.294431 -83.792628   
375 315 PHEV 42.268065 -83.719317   
810 548 PHEV 42.286983 -83.725747   
815 550 PHEV 42.287149 -83.725484   
1011 644 PHEV 42.316843 -83.699489   
... ... ... ... ...   
3695 2596 ICE 42.277079 -83.739363   
3697 2601 ICE 42.229799 -83.722897   
3721 2883 ICE 42.276460 -83.680983   
3722 2889 ICE 42.242932 -83.680755   
3723 2898 ICE 42.272489 -83.685635   
  
 Air Conditioning Power[Watts] Heater Power[Watts] Vehicle Speed[km/h] \  
2967 NaN NaN 65.298611   
375 0.0 NaN 41.429412   
810 NaN NaN 51.807823   
815 NaN NaN 57.994318   
1011 NaN NaN 69.112672   
... ... ... ...   
3695 NaN NaN 11.580198   
3697 NaN NaN 47.079208   
3721 NaN NaN 50.807860   
3722 NaN NaN 39.779528   
3723 NaN NaN 44.191071   
  
 Distance[km] Engine RPM[RPM] OAT[DegC] Generalized\_Weight FCR \  
2967 8.187884 2105.895692 -18.674603 4000.0 7.870564   
375 2.678766 498.723627 8.862685 4000.0 7.655621   
810 0.543479 2074.414966 -3.000000 4000.0 7.630475   
815 0.530423 2231.636364 7.000000 4000.0 6.844090   
1011 0.585098 1877.917431 3.614679 4000.0 6.799322   
... ... ... ... ... ...   
3695 0.674572 973.958416 NaN NaN NaN   
3697 3.337745 1591.170297 NaN 3500.0 NaN   
3721 0.796445 1235.310044 7.069869 3500.0 NaN   
3722 1.631204 1202.423622 5.344882 3500.0 NaN   
3723 6.035106 1224.420067 3.384061 3500.0 NaN   
  
 HV Battery Power[Watts] MAF[g/sec] Absolute Load[%] \  
2967 7689.712572 NaN NaN   
375 -3931.990481 0.710000 NaN   
810 6030.385734 NaN NaN   
815 6130.958314 NaN NaN   
1011 6983.277760 NaN NaN   
... ... ... ...   
3695 NaN 16.041347 22.205397   
3697 NaN NaN NaN   
3721 NaN 14.933362 28.040072   
3722 NaN 14.764094 30.331327   
3723 NaN 15.731248 32.537536   
  
 Short Term Fuel Trim Bank 1[%] Short Term Fuel Trim Bank 2[%] \  
2967 NaN NaN   
375 NaN NaN   
810 NaN NaN   
815 NaN NaN   
1011 NaN NaN   
... ... ...   
3695 NaN NaN   
3697 2.257116 NaN   
3721 -0.849481 -9.136190   
3722 -0.832923 -9.281496   
3723 -0.204783 -9.781565   
  
 Long Term Fuel Trim Bank 1[%] Long Term Fuel Trim Bank 2[%]   
2967 NaN NaN   
375 NaN NaN   
810 NaN NaN   
815 NaN NaN   
1011 NaN NaN   
... ... ...   
3695 NaN NaN   
3697 1.316522 NaN   
3721 NaN NaN   
3722 NaN NaN   
3723 NaN NaN   
  
[3727 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 42.270753 -83.730518   
2 2017-11 ICE 42.272108 -83.729198   
3 2017-11 PHEV 42.275141 -83.725549   
0 2017-11 EV 42.271939 -83.730601   
5 2017-12 HEV 42.270572 -83.728986   
6 2017-12 ICE 42.271020 -83.729605   
7 2017-12 PHEV 42.272870 -83.724669   
4 2017-12 EV 42.270083 -83.719460   
9 2018-01 HEV 42.273033 -83.730757   
10 2018-01 ICE 42.271214 -83.729482   
11 2018-01 PHEV 42.270971 -83.724416   
8 2018-01 EV 42.270827 -83.738837   
13 2018-02 HEV 42.273897 -83.731282   
14 2018-02 ICE 42.270373 -83.728213   
15 2018-02 PHEV 42.275239 -83.725741   
12 2018-02 EV 42.280399 -83.729143   
17 2018-03 HEV 42.274083 -83.730930   
19 2018-03 PHEV 42.275385 -83.721293   
18 2018-03 ICE 42.270681 -83.726283   
16 2018-03 EV 42.279812 -83.730212   
  
 Air Conditioning Power[Watts] Heater Power[Watts] Vehicle Speed[km/h] \  
1 NaN NaN 43.437104   
2 NaN NaN 37.351677   
3 104.050681 NaN 40.693952   
0 414.954458 336.061841 37.265814   
5 NaN NaN 43.244043   
6 NaN NaN 38.066903   
7 4.711894 NaN 38.266598   
4 403.627442 573.992110 48.326677   
9 NaN NaN 40.747353   
10 NaN NaN 37.099920   
11 1.523391 NaN 41.864292   
8 322.316127 1456.204868 28.852223   
13 NaN NaN 43.293240   
14 NaN NaN 37.845775   
15 13.352528 NaN 40.610221   
12 741.418041 441.426816 37.308718   
17 NaN NaN 45.993386   
19 13.900498 NaN 41.203413   
18 NaN NaN 38.830091   
16 349.696339 280.886240 41.191376   
  
 Distance[km] Engine RPM[RPM] OAT[DegC] Generalized\_Weight FCR \  
1 4.301229 1048.747374 NaN 3228.776277 0.548196   
2 4.187160 1383.917974 NaN 3549.981560 0.878507   
3 6.113691 425.540190 5.214369 3644.709011 1.062796   
0 5.296786 NaN 5.191515 3500.000000 NaN   
5 5.337657 1098.659931 NaN 3233.417042 0.562779   
6 5.011327 1395.176585 NaN 3528.523832 0.854187   
7 6.458477 531.664092 -1.216775 3616.812793 1.440610   
4 3.511447 NaN 1.345473 3500.000000 NaN   
9 5.106211 1160.830488 -3.297758 3226.098228 0.552229   
10 4.323440 1391.578431 -3.010891 3488.727476 0.825635   
11 6.139994 638.328926 -4.481358 3766.429201 2.331827   
8 5.092053 NaN -1.956272 3500.000000 NaN   
13 5.537270 1086.654282 -0.866298 3215.426621 0.549437   
14 4.299338 1393.884041 -1.087634 3530.293338 0.836752   
15 4.643482 615.558216 -2.942089 3649.323402 1.451167   
12 2.863774 NaN -0.674667 3500.000000 NaN   
17 4.905029 1080.578395 4.101265 3194.066856 0.559929   
19 5.621847 415.651537 4.470072 3673.502538 0.722956   
18 4.287977 1404.494409 4.290160 3512.259264 0.829143   
16 3.909309 NaN 4.728449 3500.000000 NaN   
  
 HV Battery Power[Watts] MAF[g/sec] Absolute Load[%] \  
1 NaN 7.999570 26.907335   
2 NaN 12.314993 30.346357   
3 -3962.267188 4.219093 NaN   
0 -5763.581448 NaN NaN   
5 NaN 8.284857 26.818614   
6 NaN 12.088989 30.830913   
7 -3341.318150 4.805669 NaN   
4 -7287.185967 NaN NaN   
9 NaN 8.226475 28.352170   
10 NaN 11.722811 31.526208   
11 -4278.804663 5.117906 NaN   
8 -6164.780102 NaN NaN   
13 NaN 8.091059 26.819778   
14 NaN 11.748619 30.973689   
15 -3485.215728 5.604098 NaN   
12 -6845.415650 NaN NaN   
17 NaN 8.201055 26.934317   
19 -4245.611082 4.409567 NaN   
18 NaN 11.549739 30.656356   
16 -6345.064556 NaN NaN   
  
 Short Term Fuel Trim Bank 1[%] Short Term Fuel Trim Bank 2[%] \  
1 -0.680006 -0.132401   
2 0.489267 0.106684   
3 NaN NaN   
0 NaN NaN   
5 -0.775959 -0.553670   
6 0.366211 -0.181285   
7 NaN NaN   
4 NaN NaN   
9 -0.698954 0.281189   
10 0.479709 0.534779   
11 NaN NaN   
8 NaN NaN   
13 -0.513524 -0.305905   
14 0.539864 -0.072874   
15 NaN NaN   
12 NaN NaN   
17 -0.618517 0.461240   
19 NaN NaN   
18 0.567216 -0.164210   
16 NaN NaN   
  
 Long Term Fuel Trim Bank 1[%] Long Term Fuel Trim Bank 2[%]   
1 0.041647 -0.494038   
2 1.526966 1.544187   
3 NaN NaN   
0 NaN NaN   
5 0.117204 -0.832392   
6 1.737142 1.472607   
7 NaN NaN   
4 NaN NaN   
9 -0.803102 -1.831730   
10 1.142479 1.572542   
11 NaN NaN   
8 NaN NaN   
13 -0.266214 -1.733216   
14 1.266370 1.324676   
15 NaN NaN   
12 NaN NaN   
17 0.196474 -1.386758   
19 NaN NaN   
18 2.044690 1.628773   
16 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 42.278267 -83.732073   
0 2017-11-01 EV 42.278107 -83.758085   
1 2017-11-01 HEV 42.271738 -83.735503   
2 2017-11-01 ICE 42.275334 -83.735704   
7 2017-11-02 PHEV 42.268268 -83.725600   
.. ... ... ... ...   
278 2018-03-27 HEV 42.271589 -83.732170   
279 2018-03-27 ICE 42.272538 -83.719731   
280 2018-03-27 PHEV 42.271897 -83.722388   
277 2018-03-27 EV 42.247692 -83.760389   
281 2018-03-28 HEV 42.281286 -83.713786   
  
 Air Conditioning Power[Watts] Heater Power[Watts] Vehicle Speed[km/h] \  
3 426.147883 NaN 34.700193   
0 55.038760 2096.899225 35.550077   
1 NaN NaN 43.912155   
2 NaN NaN 32.379000   
7 46.055131 NaN 38.199360   
.. ... ... ...   
278 NaN NaN 49.159311   
279 NaN NaN 43.773547   
280 59.212198 NaN 45.297651   
277 350.000000 0.000000 52.620438   
281 NaN NaN 57.026316   
  
 Distance[km] Engine RPM[RPM] OAT[DegC] Generalized\_Weight FCR \  
3 2.692697 142.962038 8.476688 3720.254777 0.000000   
0 0.490034 NaN 5.000000 3500.000000 NaN   
1 3.586749 1013.159684 NaN 3201.911308 NaN   
2 3.824206 1320.550910 NaN 3536.946296 NaN   
7 6.846114 476.956610 4.254588 3431.743514 0.000000   
.. ... ... ... ... ...   
278 4.280027 1100.457228 2.909523 3229.915696 0.567006   
279 3.065973 1477.459770 1.827866 3428.682365 0.895238   
280 1.533832 387.065237 2.099985 4000.000000 1.414753   
277 0.909519 NaN 1.775338 3500.000000 NaN   
281 3.065481 1385.764706 1.184211 3000.000000 0.699542   
  
 HV Battery Power[Watts] MAF[g/sec] Absolute Load[%] \  
3 -5019.208787 3.129290 NaN   
0 -4495.982558 NaN NaN   
1 NaN 7.528161 25.843640   
2 NaN 11.613017 29.339840   
7 -2729.091397 4.343766 NaN   
.. ... ... ...   
278 NaN 8.318865 26.552560   
279 NaN 12.560935 32.143458   
280 -4606.851790 2.881144 NaN   
277 -8331.605574 NaN NaN   
281 NaN 10.506053 30.960967   
  
 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   
.. ... ...   
278 -0.549761 1.600151   
279 0.909867 -3.329158   
280 NaN NaN   
277 NaN NaN   
281 -0.665151 NaN   
  
 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   
.. ... ...   
278 0.829528 -2.992021   
279 1.466398 1.496950   
280 NaN NaN   
277 NaN NaN   
281 -1.246856 NaN   
  
[282 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] \  
78881 42.237157 -83.730340 PHEV 121.953125   
178231 42.249462 -83.683875 PHEV 134.000000   
80339 42.237267 -83.726786 PHEV 120.000000   
80766 42.237298 -83.725577 PHEV 119.134375   
797285 42.289601 -83.803461 PHEV 112.480903   
... ... ... ... ...   
984100 42.325111 -83.748894 ICE 80.555556   
984115 42.325151 -83.748901 ICE 85.250000   
984159 42.325349 -83.748998 ICE 97.857143   
984161 42.325380 -83.748911 ICE 91.125000   
984171 42.325568 -83.802335 ICE 71.666667   
  
 Absolute Load[%] Engine RPM[RPM] OAT[DegC] Generalized\_Weight \  
78881 NaN 4060.800000 7.0 3500.0   
178231 NaN 3405.666667 6.0 3500.0   
80339 NaN 3947.600000 6.5 3500.0   
80766 NaN 3424.600000 6.5 3500.0   
797285 NaN 3175.666667 -18.5 4000.0   
... ... ... ... ...   
984100 NaN 1371.666667 NaN 5000.0   
984115 74.509804 1683.000000 NaN 4500.0   
984159 NaN 1640.000000 NaN 5000.0   
984161 NaN 1528.875000 NaN 5000.0   
984171 16.993465 2715.555556 NaN 3000.0   
  
 FCR HV Battery Power[Watts]   
78881 19.924206 -23907.512715   
178231 17.677546 -524.468750   
80339 17.090837 11379.409279   
80766 16.327641 -54494.025916   
797285 16.055752 7260.586510   
... ... ...   
984100 NaN NaN   
984115 NaN NaN   
984159 NaN NaN   
984161 NaN NaN   
984171 NaN NaN   
  
[984179 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'])]

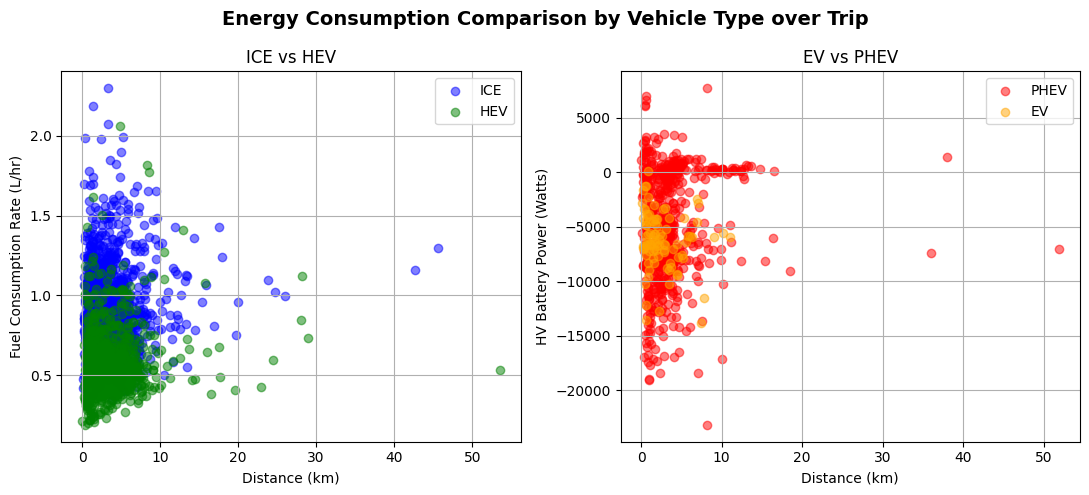
for i in df\_EC\_trip\_ICE\_HEV.columns:  
 print(f"Unique values in column '{i}':")  
 print(df\_EC\_trip\_ICE\_HEV[i].unique())  
 print(df\_EC\_trip\_ICE\_HEV[i].value\_counts())

Unique values in column 'Trip':  
[1637 293 340 ... 2883 2889 2898]  
Trip  
1473 2  
745 2  
629 2  
447 2  
1445 2  
 ..  
2512 1  
2511 1  
2506 1  
2480 1  
2471 1  
Name: count, Length: 1976, dtype: int64  
Unique values in column 'Vehicle Type':  
['ICE' 'HEV']  
Vehicle Type  
ICE 1850  
HEV 1164  
Name: count, dtype: int64  
Unique values in column 'Latitude[deg]':  
[42.27566207 42.26802927 42.26990241 ... 42.27646014 42.24293158  
 42.27248859]  
Latitude[deg]  
42.272489 1  
42.275662 1  
42.268029 1  
42.269902 1  
42.247261 1  
 ..  
42.299086 1  
42.271396 1  
42.278524 1  
42.272048 1  
42.254258 1  
Name: count, Length: 3014, dtype: int64  
Unique values in column 'Longitude[deg]':  
[-83.68216709 -83.68034 -83.69598954 ... -83.68098319 -83.68075538  
 -83.68563487]  
Longitude[deg]  
-83.685635 1  
-83.682167 1  
-83.680340 1  
-83.695990 1  
-83.698104 1  
 ..  
-83.727965 1  
-83.727527 1  
-83.739967 1  
-83.701715 1  
-83.739772 1  
Name: count, Length: 3014, 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]':  
[71.19958848 55.28043478 55.13157895 ... 50.80786026 39.77952756  
 44.19107057]  
Vehicle Speed[km/h]  
44.191071 1  
71.199588 1  
55.280435 1  
26.577075 1  
37.622549 1  
 ..  
43.955584 1  
29.430647 1  
36.728477 1  
78.166852 1  
92.660819 1  
Name: count, Length: 3011, dtype: int64  
Unique values in column 'Distance[km]':  
[3.33688203 1.33733907 3.37250506 ... 0.79644493 1.63120429 6.03510552]  
Distance[km]  
6.035106 1  
3.336882 1  
1.337339 1  
2.660535 1  
2.668459 1  
 ..  
5.009314 1  
2.389135 1  
0.398852 1  
5.286740 1  
4.808810 1  
Name: count, Length: 3011, dtype: int64  
Unique values in column 'Engine RPM[RPM]':  
[1626.54320988 1539.62173913 1394.70942982 ... 1235.31004367 1202.42362205  
 1224.42006721]  
Engine RPM[RPM]  
1224.420067 1  
1626.543210 1  
1539.621739 1  
1227.150198 1  
1293.132353 1  
 ..  
1415.465313 1  
1237.225892 1  
1340.377483 1  
1849.144605 1  
1620.105263 1  
Name: count, Length: 3011, dtype: int64  
Unique values in column 'OAT[DegC]':  
[ nan -4.95454545e+00 6.61165049e+00 1.82651072e+00  
 2.70872642e+00 8.63851852e+00 -9.00000000e+00 -2.00000000e+00  
 2.43607306e+00 2.63051471e+00 2.30936819e+00 1.80000000e+01  
 1.17212121e+01 -5.75632615e+00 6.34841629e+00 7.93333333e+00  
 6.00000000e+00 3.52595495e-02 -3.42238267e+00 2.14375000e+00  
 3.18905473e+00 6.66355763e+00 7.31372549e+00 -4.47282609e+00  
 -1.62660256e+00 5.00000000e+00 4.00000000e+00 -2.32714617e+00  
 -6.00000000e+00 8.76675603e-01 3.00000000e+00 0.00000000e+00  
 4.16964286e+00 -1.97500000e+01 -7.21229868e+00 6.89285714e+00  
 -1.96786248e+00 -1.77468840e+00 9.64166667e+00 -3.73983740e-01  
 2.00000000e+00 7.48409894e+00 2.57783019e+00 5.40531561e+00  
 3.10149942e+00 2.46734694e+00 -3.32539683e+00 1.30019531e+01  
 -1.00000000e+00 4.38941799e+00 4.84803922e+00 4.94086957e+00  
 -6.45692884e+00 -5.22727273e-01 -4.30392157e+00 1.03333333e+01  
 1.40514469e+00 2.58023020e+00 -3.62104430e+00 1.10000000e+01  
 -6.57696897e+00 2.60912548e+00 1.31822863e+00 5.97986577e+00  
 1.50000000e+01 -2.89568345e+00 1.89158345e+00 -6.51315789e-01  
 5.09347025e+00 7.28620690e+00 1.64434524e+00 -1.05611814e+01  
 -1.51658768e-01 -4.28631757e+00 1.73486020e+01 -7.38170347e-01  
 5.52334152e+00 -1.10000000e+01 8.00000000e+00 1.00000000e+00  
 -3.61702128e+00 -1.92100193e+00 9.51585977e-01 8.59885932e+00  
 -1.96054519e+00 4.20132743e+00 6.13846154e+00 7.19433198e+00  
 3.02592593e+00 3.47342398e+00 -7.70781893e+00 4.89440994e+00  
 -1.43216080e+00 -3.00000000e+00 -6.77142857e+00 2.85453034e+00  
 1.30280830e+00 6.92282958e+00 5.24689826e+00 7.59730458e+00  
 -6.76299945e+00 -8.00000000e+00 1.53550043e+00 4.85653105e+00  
 1.56948138e+00 -7.64572048e+00 7.62925170e+00 3.35339639e+00  
 1.34598930e+01 -2.71661238e+00 -7.29333333e+00 -3.77022654e+00  
 2.11111111e+00 -3.81924198e-01 -1.09466985e+00 4.60625674e-01  
 -2.11415781e+00 -5.32825719e+00 -1.09927602e+01 4.12478336e+00  
 -7.86283525e+00 -3.51898734e+00 -1.29635417e+01 1.34722524e+01  
 5.39887188e-02 6.40506329e+00 -3.47295597e+00 -5.00000000e+00  
 2.15945330e+00 -9.64285714e-01 6.66478343e+00 2.81639929e-01  
 -1.30255183e+01 6.60000000e+00 -7.00000000e+00 -7.17777778e+00  
 -1.50000000e+01 -5.58139535e+00 -3.72020725e+00 -1.03571429e+00  
 7.81515152e+00 -9.89440994e+00 -5.57740586e+00 -1.84000000e-01  
 -4.36565325e+00 7.87164179e+00 -8.80359435e+00 1.61527415e+01  
 5.72622846e+00 1.52285396e+00 -4.00000000e+00 -4.85743381e+00  
 -3.51851852e-01 4.08766486e+00 -5.97185430e+00 1.59718310e+01  
 1.52459016e+00 -4.87371512e+00 1.03700787e+01 -1.25657534e+01  
 6.49761905e+00 1.46689895e+00 8.69354839e+00 4.15607190e-01  
 -4.13473424e+00 1.73611111e+00 -5.42897196e+00 8.42988506e+00  
 2.22815534e+00 1.56883510e+00 3.59375000e-01 4.96598639e-01  
 -3.76157965e+01 1.53363961e+01 1.77755906e+00 -7.50632911e+00  
 6.26957831e+00 -5.26385809e+00 2.87694146e+00 7.00000000e+00  
 7.81269841e+00 6.17067834e+00 -6.41929499e-01 1.80117302e+00  
 -2.91855204e-01 7.05584642e+00 1.57714286e+00 -9.45241199e+00  
 4.98051948e+00 4.60302866e+00 -2.46939954e+00 6.27777778e+00  
 -1.77000000e+00 -3.25735294e-01 2.81879446e+00 2.66526020e+00  
 1.31771978e+01 1.14554674e+01 3.63000000e+00 6.50735294e+00  
 4.75895317e-01 5.88355167e+00 -6.25144509e+00 -2.66666667e+00  
 2.04996157e+00 2.84477015e-01 -1.53072871e+00 6.50071531e+00  
 -1.46199829e+00 -3.82000000e+00 2.94092827e+00 4.04054054e+00  
 -7.87022901e+00 -2.40584416e+00 6.79097387e+00 -2.80906344e+00  
 5.98347107e+00 -1.50483203e-01 1.00000000e+01 9.60471976e+00  
 2.05617978e+00 1.32029795e+00 6.66304348e+00 2.81849315e+00  
 -5.69160584e+00 3.42270862e+00 1.03987261e+01 2.60606061e+00  
 8.16524702e+00 -2.86630795e+00 1.48395722e+00 4.95254237e+00  
 1.20000000e+01 3.02266082e+00 1.91593886e+00 -4.54796031e+00  
 8.69158879e-01 -2.69041769e+00 1.04900332e+01 3.08347902e+00  
 7.18605466e+00 -6.58477509e+00 7.07951070e+00 5.79753680e-01  
 6.40067912e-01 5.09671695e+00 1.09912854e+00 -1.15732369e-01  
 -4.71764706e-01 -6.71442688e-01 6.56533828e+00 -4.08213552e+00  
 8.23956443e+00 -7.97817346e+00 -4.14385965e+00 -2.67086835e+00  
 -1.71118012e+00 8.36419753e+00 1.75309951e+01 -1.64235624e+00  
 7.66720779e+00 2.60740741e+00 4.30000000e+00 8.49926435e-01  
 -3.21224921e+00 -4.91395793e-01 -2.19858156e+00 -2.27109974e+00  
 9.00000000e+00 4.85240175e+00 3.44961240e+00 5.16385542e+00  
 -1.30000000e+01 7.50232558e+00 -5.37117904e-01 -4.41167665e+00  
 1.64713419e+01 -9.06666667e-01 7.89977974e+00 -7.47967480e+00  
 1.30000000e+01 1.35937500e+00 1.90000000e+01 2.69796709e+00  
 5.35871157e-01 -6.33116178e+00 3.55303030e+00 5.46122449e+00  
 5.07254534e+00 -4.54887984e+00 -6.33561644e+00 1.18720153e+00  
 2.54385965e-01 2.89703316e-01 6.13614341e+00 -6.11948052e-01  
 -1.83431953e-01 -8.72600349e-03 -1.09868421e+00 -1.88562904e+00  
 -1.50252525e-01 7.96366509e+00 5.52916931e+00 5.18500000e+00  
 5.25559482e+00 -7.49162011e+00 1.13644068e+01 -4.70051414e+00  
 1.50081855e+01 6.25000000e+00 4.05250875e+00 -3.42757417e+00  
 5.85114806e+00 6.73480663e+00 6.51226551e+00 -4.42376682e+00  
 2.64285714e+00 -2.49744712e+00 5.17004049e+00 -4.14142678e+00  
 -9.13043478e-01 -3.48984772e+00 4.51798561e+00 8.21317365e+00  
 -5.35494881e+00 4.86837061e+00 -3.68602645e+00 6.98479392e+00  
 7.10027599e+00 6.33977067e+00 1.58383339e+00 -2.23500000e+00  
 -6.98757764e+00 -3.60946746e+00 1.35025818e+01 5.66348656e-01  
 3.82123894e+00 6.13500649e+00 6.24871355e+00 -1.80000000e+01  
 6.31818182e-01 -1.06420233e+00 9.88771930e-01 4.45434298e-02  
 2.57771664e+00 1.13136289e+00 1.37771613e-01 5.15671642e+00  
 6.16056034e+00 -1.09675366e+00 -6.49746193e+00 5.29411765e-01  
 -9.74683544e-01 1.66480447e+00 1.37405628e+00 -1.06419237e+01  
 2.94650206e+00 3.70246914e+00 -4.89743590e+00 4.39739884e+00  
 1.23387097e+00 2.20665742e+00 7.74269006e+00 4.28336621e+00  
 6.09375000e+00 -3.54300761e+00 2.84277879e+00 -8.28678304e+00  
 -1.77076412e+00 -2.88293651e+00 -1.48438228e+01 4.81050463e+00  
 -6.34329169e+00 -9.66091954e+00 7.54804270e+00 2.28688525e-01  
 6.51779935e+00 -3.18756170e+00 3.87426901e+00 1.30172414e+00  
 -7.06521739e-02 -9.42885772e+00 -6.90756972e+00 -5.05484694e+00  
 9.21324864e+00 1.94091904e+00 6.75115473e+00 -1.00000000e+01  
 -2.57314629e+00 1.37209302e+00 2.47112861e+00 -5.22050060e+00  
 -1.05784062e+00 1.49075452e+01 5.52688172e+00 -3.57746479e-01  
 1.07623762e+01 -3.22297297e+00 2.84542816e+00 4.49106803e+00  
 7.56143667e+00 5.98385965e+00 -6.97864184e+00 1.91405765e+01  
 3.78883495e+00 -3.22962963e+00 -2.62567568e+00 2.52004860e+00  
 2.32991266e+01 1.22177419e+00 -4.59061834e+00 3.39530235e+00  
 4.26558603e+00 4.57958873e+00 1.39071856e+01 9.01702128e+00  
 8.71176471e+00 2.64928910e+00 5.16062992e+00 -3.08760331e-01  
 4.86307692e+00 1.35911330e+01 5.94840295e+00 3.10040984e+00  
 6.75519073e-01 -1.22110656e+01 3.71794872e-01 1.47053571e+01  
 1.09344043e+01 -1.78516229e+00 9.21782178e+00 3.49781182e+00  
 1.35568862e+01 2.15861718e+00 -4.73121387e+00 -7.99226306e+00  
 -3.04940120e+00 6.61016949e+00 3.75407779e+00 -1.87043796e+00  
 -1.54658385e+00 6.75052411e-01 5.42399173e-01 1.09271523e+00  
 7.62373737e+00 6.00000000e-01 4.83673469e+00 -3.48205128e+00  
 -7.89887640e+00 -6.65326633e-01 1.92943201e+01 1.00253807e+00  
 4.04132231e+00 4.53825137e+00 -5.31381872e+00 5.71052632e+00  
 -2.36865342e+00 -1.61629048e+00 1.18421053e+00 7.78727842e+00  
 -5.78909091e+00 5.70175439e-02 -2.84740883e+00 5.82741935e+00  
 6.04166667e-02 7.89944134e+00 1.12202011e+01 -2.32949640e+00  
 -1.34763948e+00 -5.29160420e+00 1.16949153e-01 3.79829242e+00  
 -9.57703927e-01 1.64404513e+01 -7.95597484e-01 5.60428850e+00  
 -6.73923304e+00 -9.32000000e+00 -3.83131923e+00 1.47629310e+00  
 5.82542694e-01 -8.18775791e+00 -1.48459384e-01 6.75438596e+00  
 5.17213115e+00 -1.93582511e+00 -4.49657869e+00 -5.39583333e+00  
 -2.72727273e-01 -9.12518854e-02 1.13828283e+01 2.04808212e+00  
 -1.21052632e+00 -5.60238429e+00 -3.25020851e+00 4.85565670e+00  
 -1.35802469e-01 8.66467463e+00 8.09547244e+00 5.29796512e+00  
 -5.50387597e-01 -2.85553772e+00 -1.62867320e+00 4.49277457e+00  
 2.23414072e+00 1.40496520e+01 7.12248322e+00 8.33767927e+00  
 6.85747126e+00 7.28061224e+00 -1.80424528e-01 -1.59770115e+00  
 1.25274725e+00 -1.43548387e+00 9.43523810e+00 -6.15584416e+00  
 7.48594378e+00 1.99522510e-01 2.98801743e+00 -3.89636570e+00  
 -7.47156827e+00 1.56382450e+01 6.34612245e+00 3.36631579e+00  
 4.15077821e+00 3.59622642e+00 9.85116279e+00 5.63833635e+00  
 2.51955307e+00 -7.52781740e+00 -2.10617284e+00 2.99443207e+00  
 1.48837209e+00 -2.65488907e+00 1.03475046e+01 5.45322581e+00  
 -7.63855422e+00 -2.33703704e+00 -1.63113006e+00 -9.43396226e-02  
 5.78633721e+00 -7.98804781e-01 -1.72273567e+00 5.37704918e+00  
 -5.02915452e+00 -3.32441472e+00 3.18452381e+00 2.09234508e+00  
 3.27124183e+00 5.59516129e+00 2.09166667e+00 2.87755102e+00  
 6.22899729e+00 5.51236631e+00 -1.12432432e+00 -6.38064516e+00  
 -7.74213836e+00 -5.16239316e+00 3.55140187e+00 2.05567929e+00  
 -3.57115750e+00 2.30601618e+01 -5.77624309e+00 -8.66310160e-01  
 1.24913793e+01 -1.26215856e+00 1.27553957e+00 -1.14430823e+01  
 1.32935780e+01 4.86723508e+00 -1.64772727e+00 3.07029973e+00  
 -6.01265823e-02 -5.52902844e+00 -3.64568345e+00 7.17188823e-01  
 5.00729927e+00 4.33943662e+00 -2.92545099e-01 9.54381085e+00  
 8.05866667e+00 1.71090909e+00 5.02311436e+00 1.89171975e+00  
 6.86111111e+00 7.64218009e-01 -5.94376309e+00 -2.09992194e-01  
 5.15267176e+00 5.52523364e+00 4.26086957e-01 5.68627451e-01  
 2.16246057e+00 1.37374517e+00 2.75346993e+00 -3.39185393e+00  
 4.95888285e+00 -2.25625000e+00 5.52317881e+00 -5.62269129e+00  
 6.48851775e+00 -4.77272727e-02 4.55073222e+00 7.09958506e+00  
 2.43841336e+00 -3.89860835e+00 4.69892473e+00 -2.10176991e+00  
 1.79025710e+00 3.49401709e+00 1.04606299e+01 -3.64532650e+00  
 1.47352113e+01 1.40000000e+01 -4.56410256e+00 5.13375796e+00  
 4.67175573e+00 6.91152263e+00 4.58213256e+00 1.69536424e+00  
 5.85291997e+00 4.43315508e+00 7.08318739e+00 1.65036675e-01  
 5.81979695e+00 -7.03846154e-01 -8.11876833e+00 1.63763066e-01  
 4.55977230e+00 6.76767677e+00 -1.29805616e+00 6.94803695e+00  
 -5.61420345e+00 5.47257053e+00 2.96912114e+00 9.72293578e+00  
 -4.47994269e+00 -2.43560606e+00 -2.19810875e+00 3.07262570e+00  
 2.36635707e+00 -1.30638298e+00 7.25502513e+00 7.98319328e+00  
 -3.75551102e+00 -6.05161290e+00 -1.52311161e+00 1.28133971e+00  
 2.86528497e+00 1.38367347e+00 3.72822823e+00 -1.30790191e-01  
 -1.34670487e+00 -1.19530245e+00 -3.58438287e+00 -3.40540541e-01  
 1.26224329e+00 7.09508882e-01 2.26770833e+00 5.34414832e+00  
 3.01620029e+00 8.30985915e-01 -6.61479592e+00 -2.44791667e-01  
 4.76303318e+00 -8.70185058e+00 7.86746988e+00 -5.14803440e+00  
 -4.40163934e+00 -5.58469945e+00 1.87019969e+00 -1.69949917e+00  
 -4.78622328e+00 -6.37582129e+00 6.87093154e+00 7.71338251e-01  
 -4.73932927e+00 5.09803922e-01 -8.62456747e+00 -4.34782609e-02  
 9.75423729e+00 -1.15642374e+01 6.59340659e-01 4.23833168e+00  
 -1.00345224e+00 1.31655874e+01 4.91662556e+00 -1.99806202e+00  
 2.19793814e+00 -1.60980810e+00 5.85616438e+00 -1.93680297e+00  
 -1.36155606e+00 -1.21807466e+00 -6.37489177e+00 -1.77918425e-01  
 4.37404580e+00 4.68609865e+00 3.83259912e+00 3.73188406e-01  
 3.79322638e+00 -5.19275701e-01 1.10432749e+01 7.71794872e+00  
 -8.12100058e-01 2.31541219e+00 -3.15195072e-01 3.71376812e+00  
 -5.46341463e+00 -6.47093023e+00 6.34090909e+00 1.83333333e+00  
 -3.61194755e+00 6.65248227e-01 3.66906475e-01 9.22689076e+00  
 1.32570945e-01 1.35081149e+00 7.15909091e-01 -2.26493109e+00  
 6.07142857e+00 2.54344392e+00 5.28319848e-01 6.15384615e-02  
 6.39931153e+00 7.41645885e+00 5.38785626e+00 4.07154213e+00  
 6.87956811e+00 1.32561983e+01 -3.33126935e+00 5.28310168e+00  
 -2.41156654e+00 5.40647482e+00 7.15792129e+00 -1.20540541e+00  
 -4.53132832e+00 -2.79026217e+00 1.46848958e+01 2.74919614e+00  
 4.75636364e+00 -5.18194070e-01 4.85284281e+00 -2.06349206e+00  
 6.38705234e+00 2.15628971e+00 1.98491879e+00 3.05734767e+00  
 -1.51857835e+00 2.42071611e+00 -6.63815789e+00 4.26148970e+00  
 -2.81504065e+00 -2.22418136e+00 7.68707483e-01 6.47446976e+00  
 -8.88728702e+00 1.39698630e+01 1.95819936e+00 6.25722983e+00  
 3.62085308e+00 8.26659292e+00 -6.60434783e+00 6.73239437e-01  
 -9.55637255e+00 1.58139535e+00 6.43594306e+00 -2.32938856e+00  
 1.24497992e-01 -2.23529412e+00 3.00865801e+00 5.90357143e+00  
 3.95563771e+00 2.52643172e+00 -3.78733265e+00 -3.16405667e+00  
 4.29327453e+00 2.78063410e+00 1.25217391e+00 2.21480879e+00  
 -5.15447154e+00 8.55828221e+00 5.10426540e+00 6.75021533e+00  
 -1.95862765e+00 -6.14606742e+00 4.16531165e+00 3.30143541e-01  
 -9.58333333e-01 -5.71428571e-01 3.32871653e+00 -7.77655678e+00  
 -1.84491115e+00 -2.48446746e+00 6.89637306e+00 2.89485459e+00  
 2.93282443e+00 9.95136187e-01 -1.82931727e+00 5.54574132e+00  
 -2.75084175e+00 -9.26867031e+00 1.35370370e+01 -7.66531440e+00  
 -5.44131455e+00 1.66040956e+00 9.83625731e+00 2.37931034e+00  
 1.94767442e+00 4.23133333e+00 -7.34556575e+00 1.38783186e+01  
 4.04356846e+00 1.34454545e+01 -2.80000000e-01 -1.96465969e+00  
 3.24393064e+00 -2.12116136e+00 8.19350474e+00 -5.40000000e-01  
 1.39503106e+00 3.24516129e+00 1.40643034e+00 -6.34343434e+00  
 1.06627018e+00 -2.09470990e+00 1.46627358e+01 4.27626459e+00  
 3.64979757e+00 -1.98501873e+00 3.33698903e+00 -1.87500000e+00  
 5.24731183e+00 3.05203136e+00 -2.03906250e+00 7.35533333e+00  
 3.78809869e-01 -3.90558036e+00 -7.48235294e+00 2.38073394e+00  
 -6.84792627e-01 3.95230263e+00 2.85356696e-01 7.11250000e+00  
 -2.57000000e+00 4.23300971e+00 -6.06341463e+00 1.73147541e+01  
 1.19703125e+01 -2.44668934e+00 1.02328767e+01 -6.02953586e+00  
 -2.97409326e-01 -2.39276316e+00 1.36986971e+01 4.11818182e+00  
 1.99354839e+00 6.10599078e+00 -1.46688207e+00 1.66021505e+00  
 3.16666667e+00 4.71976150e+00 5.66464758e+00 4.39013453e+00  
 7.30372493e+00 2.72395833e+00 4.48780488e+00 9.36170213e-01  
 -2.90494297e+00 3.89375000e+00 3.62622951e+00 -1.55364807e+00  
 1.32812500e+00 5.60000000e+00 3.76296296e+00 1.61194030e-01  
 2.82723577e+00 -1.39935065e+00 -1.13132221e+01 8.18426104e+00  
 1.18075472e+01 1.11862245e+00 5.57942238e+00 6.62350598e+00  
 -6.83619345e+00 -2.94736842e-01 1.06719368e+00 1.53153527e+01  
 8.94868586e-02 5.81502086e+00 4.74631751e+00 7.06986900e+00  
 5.34488189e+00 3.38406145e+00]  
OAT[DegC]  
 0.000000 34  
-2.000000 30  
 3.000000 27  
 6.000000 25  
 2.000000 23  
 ..  
 5.815021 1  
 4.746318 1  
 7.069869 1  
 5.344882 1  
-4.954545 1  
Name: count, Length: 893, dtype: int64  
Unique values in column 'Generalized\_Weight':  
[5000. 4000. 4500. ... 3093.85113269 3924.44444444  
 3924.89568846]  
Generalized\_Weight  
3000.000000 602  
3500.000000 450  
2500.000000 149  
4000.000000 122  
4500.000000 103  
 ...   
3484.883721 1  
3786.956522 1  
3384.439359 1  
3645.983646 1  
3101.970865 1  
Name: count, Length: 1464, dtype: int64  
Unique values in column 'FCR':  
[2.298379 2.18847699 2.07601606 ... 0.1994697 0.18827951 nan]  
FCR  
0.263086 1  
0.268363 1  
0.268604 1  
0.279921 1  
0.281715 1  
 ..  
1.994577 1  
2.062937 1  
2.076016 1  
2.188477 1  
2.298379 1  
Name: count, Length: 2720, dtype: int64  
Unique values in column 'HV Battery Power[Watts]':  
[nan]  
Series([], Name: count, dtype: int64)  
Unique values in column 'MAF[g/sec]':  
[30.62242803 15.1794252 nan ... 14.9333623 14.7640942  
 15.73124784]  
MAF[g/sec]  
31.161226 1  
16.892765 1  
24.812324 1  
16.813280 1  
20.410097 1  
 ..  
27.862649 1  
27.939077 1  
29.435711 1  
15.179425 1  
30.622428 1  
Name: count, Length: 2800, dtype: int64  
Unique values in column 'Absolute Load[%]':  
[ nan 39.0213135 37.68274904 ... 28.04007231 30.33132675  
 32.53753589]  
Absolute Load[%]  
32.537536 1  
39.021314 1  
37.682749 1  
40.230861 1  
32.355825 1  
 ..  
35.465044 1  
29.134287 1  
30.257789 1  
29.634005 1  
35.462928 1  
Name: count, Length: 2852, dtype: int64  
Unique values in column 'Short Term Fuel Trim Bank 1[%]':  
[-0.53851595 -0.21853147 0.71081831 ... -0.84948144 -0.83292323  
 -0.20478277]  
Short Term Fuel Trim Bank 1[%]  
-0.878906 3  
-1.546224 2  
-0.488281 2  
-1.422276 2  
-0.075793 2  
 ..  
 0.722572 1  
-0.016931 1  
 0.331615 1  
-2.654474 1  
-0.204783 1  
Name: count, Length: 2853, dtype: int64  
Unique values in column 'Short Term Fuel Trim Bank 2[%]':  
[-0.2973894 -0.04916958 -1.09981796 ... -9.13618996 -9.28149606  
 -9.78156505]  
Short Term Fuel Trim Bank 2[%]  
-1.069079 2  
 0.000000 2  
-0.094499 1  
 1.011675 1  
-82.059717 1  
 ..  
-2.687312 1  
-0.738284 1  
-0.966283 1  
-0.449320 1  
-2.095406 1  
Name: count, Length: 1111, dtype: int64  
Unique values in column 'Long Term Fuel Trim Bank 1[%]':  
[10.40059156 2.76715472 1.56900139 ... 1.42385563 0.89328973  
 1.31652228]  
Long Term Fuel Trim Bank 1[%]  
-0.781250 16  
 0.000000 12  
 1.562500 11  
-2.343750 8  
-1.562500 8  
 ..  
-2.163599 1  
-0.991230 1  
 1.468129 1  
 1.108285 1  
 1.310436 1  
Name: count, Length: 2761, dtype: int64  
Unique values in column 'Long Term Fuel Trim Bank 2[%]':  
[10.13052984 -1.34670017 0.02275485 ... -3.01284035 -0.40897162  
 -3.27190171]  
Long Term Fuel Trim Bank 2[%]  
 0.000000 14  
 3.125000 10  
 1.562500 6  
 2.343750 4  
 4.687500 3  
 ..  
 7.246767 1  
-0.911074 1  
 1.425149 1  
 1.253743 1  
-3.271902 1  
Name: count, Length: 1044, dtype: int64

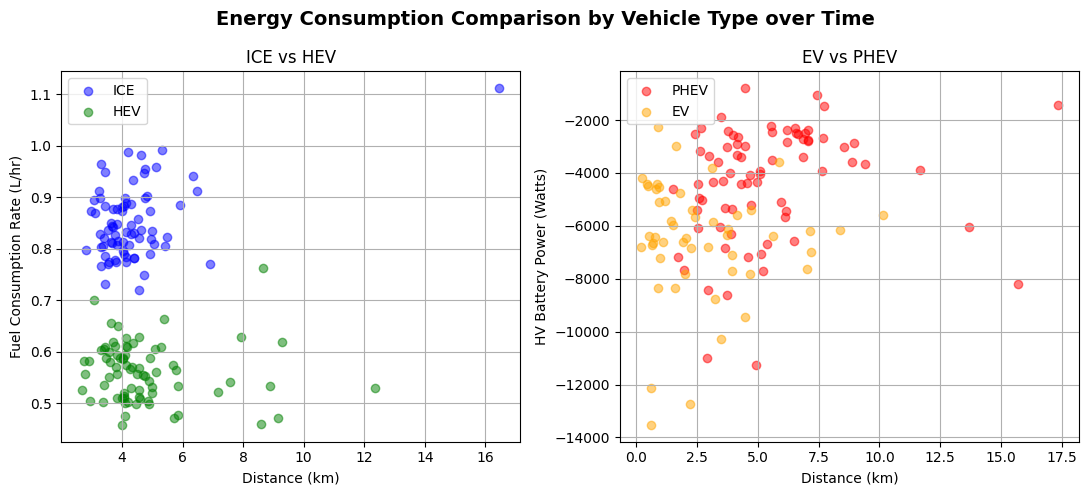
for i in df\_EC\_trip\_EV\_PHEV.columns:  
 print(f"Unique values in column '{i}':")  
 print(df\_EC\_trip\_EV\_PHEV[i].unique())  
 print(df\_EC\_trip\_EV\_PHEV[i].value\_counts())

Unique values in column 'Trip':  
[1565 315 548 550 644 499 276 485 340 643 1196 387 448 1634  
 322 513 267 446 1212 1366 498 1569 1085 1109 1011 520 750 487  
 511 281 845 278 515 1566 1003 444 453 1240 2265 552 579 451  
 1558 570 268 889 555 284 509 460 1465 335 271 512 1040 1620  
 1269 646 1029 358 648 459 305 261 842 273 1385 383 1069 1179  
 458 1072 1009 1232 625 1420 1075 1550 1199 1024 1122 569 747 599  
 1206 755 784 962 1469 1595 1214 741 285 1747 1139 742 699 1605  
 1185 456 1392 1819 327 413 793 947 664 814 493 785 688 938  
 370 488 1001 1091 1689 1816 1005 1615 1464 1598 881 1490 225 1825  
 1123 1481 539 316 702 1268 1015 890 751 1149 671 1037 1419 880  
 598 306 753 761 1247 1619 1648 954 737 1432 2413 473 790 723  
 1065 364 532 424 1451 1374 1610 678 1250 1416 1133 1282 1371 491  
 1084 472 1467 1076 1267 1438 462 710 1047 1376 1290 953 542 1265  
 259 1010 989 1278 2254 258 1414 481 1652 1226 1617 494 2246 478  
 1380 1459 589 1389 321 1653 1446 1562 1442 680 1609 634 1129 428  
 1599 502 681 1415 1216 1618 1382 1031 1187 1623 1390 1089 873 1606  
 1198 1638 629 1213 1584 612 1680 477 770 1820 489 1710 486 672  
 8 1418 709 882 120 859 382 789 1070 1243 43 769 106 663  
 1016 875 138 1731 83 122 111 637 326 280 2016 85 133 146  
 82 496 601 929 113 105 99 136 799 1012 412 416 110 368  
 135 4 1492 1560 1150 139 852 987 675 117 116 893 92 337  
 497 2234 325 44 418 1088 98 324 1041 611 415 89 795 38  
 109 379 339 14 482 336 554 1485 942 585 849 484 134 1670  
 39 1082 1701 42 1583 132 2 1401 1781 11 137 12 112 2133  
 865 1259 1248 922 820 828 150 708 1692 148 733 2420 2376 1020  
 13 1626 857 1949 34 1062 1736 884 1422 1030 538 711 826 377  
 7 863 647 18 545 1210 900 621 597 864 1143 565 1631 1908  
 529 944 667 1174 1175 797 141 1474 1911 907 821 606 572 812  
 466 1100 1168 669 1408 1582 417 1745 600 1704 719 2411 921 1585  
 1454 685 1018 1092 739 1954 1602 936 745 2432 593 662 762 935  
 1817 846 1160 705 861 1165 568 1006 1542 958 1357 1530 603 1336  
 1090 1462 659 650 1147 2036 715 304 690 695 2202 1568 1026 1578  
 2035 2379 791 1439 1994 636 940 651 1296 661 1951 642 748 2030  
 2378 1740 752 1501 1064 575 653 1386 1793 1567 937 933 2144 2028  
 1087 1561 1073 1154 803 1808 1814 1303 1343 1285 1261 588 1158 1370  
 1961 1288 1528 665 757 1335 1497 1067 2139 1914 2109 923 1897 1302  
 698 328 2021 1693 2367 1095 916 1541 1263 912 1172 1262 407 2161  
 506 1572 2034 941 1875 1086 633 2230 674 1103 1654 1889 1804 1799  
 1551 390 743 1711 1878 1882 1519 2150 673 1311 1321 915 433 694  
 2018 2105 707 510 2020 906 1128 1686 1574 1734 2247 1883 736 1449  
 1448 1601 1891 966 1962 1503 2032 1906 2029 1297 2217 1509 668 1992  
 2146 445 831 894 1173 957 319 1690 895 783 1537 2000 2165 825  
 1081 2115 1671 519 1887 1167 892 1722 1729 1308 934 2091 2244 2143  
 2121 817 768 1543 744 1007 1871 1098 2227 1178 2148 802 1725 738  
 1495 1373 1349 756 1755 1760 1904 1886 1765 735 2019 1660 687 827  
 759 918 1885 1870 1434 1719 2250 2223 1177 776 787 779 2240 1164  
 658 2203 1450 823 806 728 641 991 655 1900 1674 948 816 951  
 323 1022 819 596 684 692 1579 1758 1125 960 869 999 1678 1387  
 1352 2257 706 1779]  
Trip  
702 2  
672 2  
755 2  
678 2  
2133 2  
 ..  
1387 1  
1352 1  
2257 1  
706 1  
315 1  
Name: count, Length: 676, dtype: int64  
Unique values in column 'Vehicle Type':  
['PHEV' 'EV']  
Vehicle Type  
PHEV 618  
EV 95  
Name: count, dtype: int64  
Unique values in column 'Latitude[deg]':  
[42.2944309 42.26806549 42.28698299 42.28714852 42.31684268 42.28499925  
 42.29688467 42.26004585 42.2952593 42.27188747 42.29416664 42.25195609  
 42.28704973 42.29400682 42.28254485 42.28503904 42.28143159 42.28396238  
 42.27915472 42.29399415 42.26714868 42.27735764 42.3043819 42.23367802  
 42.26697439 42.28693085 42.2597161 42.28021774 42.2845939 42.27563651  
 42.26039099 42.26624775 42.27959483 42.26662775 42.29745574 42.28298235  
 42.30533915 42.25200339 42.24165036 42.30404118 42.2877325 42.26775685  
 42.27066216 42.26073919 42.27735129 42.28942452 42.24423774 42.28696306  
 42.29649759 42.29017718 42.2893822 42.25564838 42.2840444 42.2868803  
 42.29269027 42.25552282 42.27093232 42.24727529 42.30432869 42.27282753  
 42.29433504 42.29773692 42.31246361 42.28422799 42.29996942 42.27814631  
 42.27807034 42.27419927 42.2914717 42.25010024 42.28789337 42.24495324  
 42.28020702 42.24521924 42.28963529 42.28660838 42.27377 42.24954629  
 42.28631686 42.26335798 42.28215851 42.27818692 42.28846566 42.26629946  
 42.26776061 42.30558885 42.26948337 42.30975722 42.25139946 42.26969657  
 42.26965586 42.2907266 42.29794036 42.31031851 42.3032301 42.29796334  
 42.29052998 42.30030583 42.29309112 42.28753531 42.28567977 42.26062229  
 42.28685594 42.28223911 42.2605402 42.24666935 42.29043904 42.28142816  
 42.28784674 42.2663768 42.27384264 42.26269289 42.2814968 42.28594535  
 42.26930397 42.29092346 42.27669351 42.26972551 42.26948883 42.26951446  
 42.28623481 42.27759359 42.27373849 42.27101643 42.29118578 42.27697716  
 42.28713459 42.26985298 42.29957541 42.31284212 42.29378497 42.29592468  
 42.26692496 42.25386661 42.25835517 42.29416011 42.25738044 42.27464643  
 42.28433909 42.27353966 42.27859763 42.28940357 42.31396337 42.26918936  
 42.2924135 42.26575102 42.26675269 42.2837895 42.27762991 42.28009134  
 42.27234256 42.24038216 42.26265572 42.27608794 42.24308397 42.25546692  
 42.29358584 42.29342074 42.28615507 42.25805767 42.26647941 42.29953281  
 42.31289876 42.26526089 42.27192877 42.2468444 42.28523062 42.29642862  
 42.2769677 42.25833725 42.27838177 42.24544714 42.26379395 42.28205816  
 42.29028175 42.29398214 42.29253894 42.26758386 42.25977292 42.28404853  
 42.30417687 42.27181701 42.25592895 42.29497098 42.28780454 42.24676249  
 42.24124268 42.25234109 42.31318688 42.27503897 42.30513719 42.30912653  
 42.2788228 42.29357831 42.24015385 42.30199209 42.29362836 42.29332477  
 42.30055371 42.27895738 42.29919044 42.27296356 42.2953382 42.25965095  
 42.31211621 42.27870446 42.29366558 42.29552163 42.30885139 42.27809593  
 42.3005864 42.27870233 42.24501406 42.28152705 42.26068989 42.27773126  
 42.29386907 42.30080801 42.29903145 42.27834446 42.27916694 42.27073414  
 42.28761184 42.26374205 42.28053834 42.27917609 42.29787178 42.27722891  
 42.25819721 42.23402732 42.23180204 42.25372668 42.24089731 42.27333439  
 42.28148778 42.27327892 42.29060874 42.27509049 42.28250454 42.27610508  
 42.25080851 42.2517505 42.28271596 42.25101901 42.2623288 42.27895498  
 42.2839779 42.24286495 42.27314222 42.22977768 42.28225484 42.27215861  
 42.27345662 42.30050308 42.27464386 42.28496293 42.28847174 42.28122292  
 42.28104356 42.27967604 42.28228489 42.28148758 42.27491489 42.28133143  
 42.29331235 42.28059864 42.2811147 42.26237618 42.28195426 42.2417366  
 42.28246106 42.28214448 42.28106964 42.2817641 42.23040771 42.26170243  
 42.27978853 42.27631994 42.2821384 42.28152045 42.28956256 42.28088421  
 42.25733913 42.30247498 42.30397129 42.28526269 42.23049097 42.26136962  
 42.23092524 42.27991069 42.28191955 42.25731811 42.28178543 42.2826957  
 42.28201833 42.2781671 42.2818783 42.28166373 42.28181077 42.2496304  
 42.2872344 42.28565704 42.23041075 42.26342789 42.28207752 42.28138557  
 42.2581142 42.28163228 42.25801092 42.27768243 42.27657343 42.28221736  
 42.28170494 42.28157198 42.24407059 42.3117314 42.25970309 42.2888482  
 42.25810102 42.28225155 42.27540834 42.2924518 42.2820457 42.28400074  
 42.26360804 42.28155379 42.25735055 42.28179606 42.26871497 42.2464784  
 42.29027619 42.28014244 42.28143665 42.27984865 42.28144977 42.26167316  
 42.23545792 42.27820465 42.28476448 42.23063531 42.26733202 42.25879432  
 42.28137071 42.24972775 42.25574013 42.25336298 42.28701067 42.28139503  
 42.25936387 42.23819926 42.28220494 42.2460412 42.25821072 42.23836488  
 42.28229157 42.25646605 42.26212588 42.26597656 42.30804445 42.23615979  
 42.27339324 42.27630015 42.2589523 42.28163785 42.28241167 42.28311096  
 42.24213515 42.28221705 42.30055848 42.28612808 42.25646709 42.27127156  
 42.2698586 42.28612165 42.24439739 42.2431015 42.30646786 42.31903217  
 42.25690728 42.28420465 42.25652454 42.24606448 42.25479968 42.23054189  
 42.28173565 42.30204431 42.29357405 42.27238456 42.27914894 42.27172792  
 42.30241724 42.27665964 42.28057842 42.27254585 42.29018067 42.2502215  
 42.26414236 42.27604322 42.27589596 42.25876189 42.24538006 42.26106323  
 42.3008364 42.2445081 42.27975726 42.26927876 42.27955368 42.27784343  
 42.2679206 42.24520809 42.24929518 42.29517982 42.2689878 42.30184454  
 42.28149928 42.24571469 42.27375526 42.29288884 42.29219529 42.29540668  
 42.24900198 42.29492802 42.27640847 42.24397188 42.26138719 42.25677177  
 42.26830915 42.2472918 42.25478162 42.24325043 42.26383948 42.26062812  
 42.28202645 42.28219711 42.28092435 42.23139421 42.28455218 42.27630054  
 42.27172783 42.25917351 42.26497514 42.25940973 42.27810749 42.24616116  
 42.25626051 42.26443848 42.28477523 42.25088698 42.26583483 42.2574318  
 42.2973321 42.28397508 42.2633822 42.25856359 42.25748431 42.28849264  
 42.28105748 42.25955925 42.27902156 42.28701375 42.29053578 42.26726302  
 42.27858656 42.28805459 42.27318015 42.23395949 42.28596586 42.25593517  
 42.25630653 42.293135 42.25559002 42.24387756 42.25488576 42.28473569  
 42.28591184 42.27429422 42.29319478 42.26573914 42.29318544 42.27500287  
 42.28118489 42.28057372 42.28179856 42.2657632 42.27042217 42.28663564  
 42.28183245 42.25336978 42.2831421 42.30746468 42.24298565 42.24332399  
 42.25341378 42.27551337 42.26076836 42.2499197 42.25803554 42.29434149  
 42.26519585 42.28557885 42.27157852 42.28189331 42.28408411 42.28108028  
 42.28166626 42.26150215 42.26091873 42.2802961 42.23939336 42.27372118  
 42.26548435 42.2394058 42.27712742 42.28013149 42.28937807 42.29374021  
 42.27459388 42.29004609 42.28390289 42.25425276 42.31331122 42.28935835  
 42.27846208 42.27221936 42.26203375 42.2885879 42.2615596 42.27278201  
 42.2473357 42.30545087 42.26432245 42.27322355 42.28589816 42.28339047  
 42.28980592 42.24798016 42.28104181 42.25401158 42.2562115 42.27361721  
 42.30419551 42.28158019 42.31014127 42.2818236 42.27855593 42.25948722  
 42.27334715 42.28683575 42.25075481 42.24460602 42.2812556 42.30288969  
 42.27096981 42.27235067 42.28074768 42.27165196 42.28569259 42.27331788  
 42.27173198 42.28122905 42.26803654 42.28708181 42.27015678 42.2729922  
 42.28184578 42.23561684 42.23037583 42.25543311 42.27879357 42.25991569  
 42.25743801 42.3017359 42.27286064 42.2964887 42.26703763 42.31595571  
 42.28215321 42.27980384 42.26790769 42.26524004 42.28095563 42.25881547  
 42.2819743 42.27988406 42.26159117 42.29245105 42.28062716 42.31440989  
 42.2959372 42.28678467 42.27105469 42.27311035 42.27391156 42.2448829  
 42.28723306 42.26073025 42.28776419 42.25366642 42.26924234 42.25695518  
 42.27269121 42.25727369 42.30976928 42.24593441 42.24237037 42.30205554  
 42.27347086 42.24068688 42.28597384 42.27249952 42.24769185 42.26647726  
 42.24606795 42.27287783 42.29315553 42.25307753 42.27572697 42.26656756  
 42.28319339 42.28208657 42.28533984 42.28028957 42.2371483 42.24062123  
 42.30369345 42.23246597 42.28651542 42.27273859 42.29011303 42.23013627  
 42.28705399 42.28389977 42.29381224 42.24059935 42.3184919 42.24195202  
 42.26282407 42.29159417 42.23483882 42.27226939 42.24133132 42.2600673  
 42.2814567 42.28193718 42.27161457 42.28245211 42.26323439 42.29350267  
 42.28277955 42.26289319 42.23733935 42.275569 42.28144148 42.28210562  
 42.26083964 42.27332982 42.31901441 42.25770968 42.25985277 42.30639035  
 42.28670581 42.26312131 42.25038273 42.24362584 42.25763982 42.24133935  
 42.25206438 42.26364713 42.27270336 42.26066695 42.267986 42.28153407  
 42.31250458 42.24320078 42.24447006 42.24216937 42.2822874 42.27586665  
 42.24231093 42.26485301 42.27532357 42.23696152 42.25894615 42.29222461  
 42.24396524 42.24262826 42.25196682 42.27032443 42.30682793 42.26180963  
 42.23344624 42.29219546 42.25680097 42.24562246 42.26417546 42.24189651  
 42.2539878 42.24417276 42.26445006 42.23043065 42.24202631 42.26352419  
 42.28945042 42.24622419 42.29646371 42.25788801 42.29479638 42.26732008  
 42.24085638 42.2819667 42.26028966 42.23153164 42.2633215 ]  
Latitude[deg]  
42.263322 1  
42.294431 1  
42.268065 1  
42.286983 1  
42.287149 1  
 ..  
42.294007 1  
42.287050 1  
42.251956 1  
42.294167 1  
42.271887 1  
Name: count, Length: 713, dtype: int64  
Unique values in column 'Longitude[deg]':  
[-83.79262824 -83.71931677 -83.7257473 -83.72548448 -83.69948867  
 -83.73501167 -83.693271 -83.72937026 -83.72212816 -83.7056074  
 -83.69612058 -83.70123589 -83.72518861 -83.69727747 -83.76293404  
 -83.73447135 -83.76681921 -83.73491547 -83.6811567 -83.69755538  
 -83.70239476 -83.73141925 -83.71534238 -83.7146541 -83.73454897  
 -83.72732082 -83.71511345 -83.71641991 -83.73528419 -83.70626804  
 -83.74549447 -83.71607046 -83.73407459 -83.75523559 -83.71795452  
 -83.70574804 -83.71040331 -83.68327752 -83.70378394 -83.70087803  
 -83.73056829 -83.71623717 -83.72188474 -83.73085314 -83.77876667  
 -83.73666039 -83.72112515 -83.72520815 -83.72550791 -83.68223043  
 -83.72935047 -83.71691487 -83.73698232 -83.72626852 -83.69488726  
 -83.74725738 -83.73682904 -83.73032493 -83.70256874 -83.71472129  
 -83.71373193 -83.7859149 -83.71386073 -83.73498963 -83.79010326  
 -83.73446075 -83.76346983 -83.70417846 -83.74117045 -83.73708619  
 -83.68533859 -83.72556654 -83.70107622 -83.75841005 -83.74969143  
 -83.7427606 -83.73664328 -83.71469606 -83.71853494 -83.76862617  
 -83.77579962 -83.7372307 -83.75405102 -83.70803512 -83.76290043  
 -83.70140748 -83.75901967 -83.73516364 -83.70307547 -83.7500729  
 -83.77249836 -83.70771089 -83.70476848 -83.67809611 -83.7026202  
 -83.72136806 -83.74156357 -83.73548196 -83.69452924 -83.72327563  
 -83.79880033 -83.73074236 -83.72081646 -83.7193987 -83.74555093  
 -83.73524254 -83.75577107 -83.75291197 -83.70980146 -83.76938314  
 -83.74382484 -83.73371045 -83.73379711 -83.70149182 -83.75089972  
 -83.73427525 -83.75436326 -83.750324 -83.74842523 -83.7503559  
 -83.7079896 -83.74312988 -83.75942206 -83.74549329 -83.7407986  
 -83.77756706 -83.72600293 -83.74884219 -83.73483986 -83.70734894  
 -83.69681934 -83.69651524 -83.72777599 -83.75466834 -83.73161281  
 -83.72906911 -83.72301619 -83.79038242 -83.74607929 -83.76596107  
 -83.68209993 -83.74882762 -83.70945584 -83.74811978 -83.72772102  
 -83.76193299 -83.75448043 -83.77120523 -83.7312437 -83.76270226  
 -83.71877953 -83.70190815 -83.72887585 -83.75588245 -83.6827483  
 -83.73996247 -83.69633344 -83.69612962 -83.71430066 -83.75836754  
 -83.75303108 -83.7129308 -83.70751129 -83.74918326 -83.74668428  
 -83.72502809 -83.71170064 -83.72521075 -83.68071139 -83.70004787  
 -83.68087836 -83.72572731 -83.73190257 -83.75424656 -83.72225458  
 -83.69664659 -83.73717985 -83.7579436 -83.7400237 -83.76362821  
 -83.69996289 -83.72192564 -83.69085729 -83.72793761 -83.70194682  
 -83.69828544 -83.7030047 -83.69269497 -83.70657277 -83.70040042  
 -83.70394762 -83.68961535 -83.69078168 -83.69680202 -83.68256571  
 -83.71709368 -83.69543067 -83.69498589 -83.67962094 -83.68100722  
 -83.79500864 -83.68098078 -83.71478654 -83.73398023 -83.70753512  
 -83.6805049 -83.69580784 -83.68216905 -83.70731303 -83.68065368  
 -83.6794177 -83.68069927 -83.70252112 -83.76170932 -83.72433616  
 -83.68042588 -83.69600584 -83.67935101 -83.67987334 -83.68166196  
 -83.68171279 -83.68076379 -83.79961021 -83.69713391 -83.6816569  
 -83.68068598 -83.68728362 -83.68013088 -83.70252891 -83.7390415  
 -83.73887824 -83.68018799 -83.68528906 -83.74005975 -83.71455647  
 -83.78638655 -83.70916056 -83.7713033 -83.71337662 -83.74667204  
 -83.7177135 -83.74908072 -83.72095101 -83.72512615 -83.73209903  
 -83.77201388 -83.75018036 -83.7206845 -83.71173712 -83.73079613  
 -83.71759574 -83.75685039 -83.75639564 -83.7467167 -83.71748124  
 -83.68484655 -83.71569994 -83.7243515 -83.72048452 -83.75114704  
 -83.71171369 -83.71787143 -83.71704226 -83.71814608 -83.71054658  
 -83.73651364 -83.6948519 -83.71427571 -83.77639554 -83.73037327  
 -83.71690988 -83.71647326 -83.72561474 -83.72277459 -83.73542493  
 -83.73991875 -83.71673707 -83.73225399 -83.71855807 -83.71881556  
 -83.71297689 -83.72270526 -83.72412564 -83.73404408 -83.69315567  
 -83.71350922 -83.68803794 -83.73046341 -83.73598273 -83.73638717  
 -83.71790581 -83.71031075 -83.71893794 -83.72459621 -83.7163512  
 -83.69380016 -83.7217211 -83.71774089 -83.71477128 -83.73968664  
 -83.71485652 -83.73435838 -83.69050738 -83.77146317 -83.71687809  
 -83.7193165 -83.75206214 -83.71955569 -83.70501741 -83.71881793  
 -83.7365162 -83.7206869 -83.71762944 -83.70969479 -83.73664374  
 -83.73050902 -83.7267854 -83.74072179 -83.74289066 -83.71806527  
 -83.71951007 -83.77499694 -83.71810658 -83.75140177 -83.74525052  
 -83.70988089 -83.71266777 -83.71588271 -83.71803327 -83.7002505  
 -83.73731609 -83.71639912 -83.71898545 -83.71361181 -83.71676993  
 -83.70610185 -83.71582432 -83.75519876 -83.75849639 -83.69049462  
 -83.79695357 -83.73968798 -83.71370252 -83.72439847 -83.68909802  
 -83.75544014 -83.75754749 -83.708146 -83.71496033 -83.72493243  
 -83.71647045 -83.69933454 -83.72485288 -83.71886346 -83.71522172  
 -83.70997287 -83.70610256 -83.75497084 -83.6875838 -83.71725088  
 -83.77453004 -83.73943416 -83.73961979 -83.71810103 -83.71632553  
 -83.74963996 -83.73196486 -83.71115068 -83.74036926 -83.75895793  
 -83.70976884 -83.79650404 -83.72296784 -83.72920813 -83.73192227  
 -83.72147801 -83.68764726 -83.71667866 -83.70949116 -83.77281923  
 -83.71268972 -83.72130658 -83.72507451 -83.68826465 -83.71239416  
 -83.70591892 -83.7312583 -83.71538017 -83.74867145 -83.77427483  
 -83.70358786 -83.7620796 -83.77852851 -83.74921538 -83.74344072  
 -83.74711556 -83.75417284 -83.74703403 -83.75605648 -83.73652951  
 -83.71850673 -83.75113422 -83.78085514 -83.71888174 -83.69965718  
 -83.72653825 -83.71288663 -83.69412741 -83.71901066 -83.73212507  
 -83.72010657 -83.72622475 -83.75011935 -83.70900344 -83.74477144  
 -83.7182277 -83.75663421 -83.732985 -83.74761529 -83.74430363  
 -83.72234284 -83.77293636 -83.766701 -83.73616611 -83.73824817  
 -83.7243346 -83.72710301 -83.72382339 -83.72499266 -83.69710427  
 -83.75472716 -83.74957987 -83.71915141 -83.7794315 -83.70732696  
 -83.72442258 -83.73680356 -83.79241798 -83.7027532 -83.73872815  
 -83.70399047 -83.71296926 -83.75808505 -83.76267072 -83.7399738  
 -83.72344708 -83.72945279 -83.70471981 -83.73530117 -83.74058542  
 -83.70983824 -83.74329906 -83.79842814 -83.73699628 -83.71033286  
 -83.68773079 -83.74544925 -83.71218254 -83.75155446 -83.73820897  
 -83.68284876 -83.73836572 -83.71244506 -83.71328864 -83.75640656  
 -83.73817348 -83.73679046 -83.69186235 -83.70986623 -83.73367358  
 -83.70987988 -83.72954492 -83.70577251 -83.72134507 -83.6899413  
 -83.74408312 -83.74051544 -83.73482994 -83.72542857 -83.75335171  
 -83.74552585 -83.73861389 -83.74057543 -83.73544794 -83.72101804  
 -83.73885034 -83.71869265 -83.69771305 -83.74321634 -83.71452036  
 -83.72041343 -83.71884677 -83.75789496 -83.77009473 -83.74022629  
 -83.69737307 -83.71167711 -83.72340099 -83.69612429 -83.72112269  
 -83.71900539 -83.71909156 -83.74577605 -83.72173617 -83.70735212  
 -83.74017688 -83.74681783 -83.70294482 -83.73515373 -83.70992091  
 -83.69568831 -83.73890211 -83.70146297 -83.73651077 -83.71297783  
 -83.77149008 -83.77723198 -83.69306262 -83.7369816 -83.70888778  
 -83.70852567 -83.74137866 -83.71276543 -83.75203137 -83.74245953  
 -83.72142583 -83.70296826 -83.74283119 -83.73721371 -83.6911744  
 -83.70333466 -83.78827868 -83.7173775 -83.74289356 -83.72201397  
 -83.70027246 -83.7085822 -83.69811966 -83.70994492 -83.73126291  
 -83.69715876 -83.75463764 -83.70832943 -83.7080205 -83.77924778  
 -83.75262518 -83.71017248 -83.71154342 -83.70993745 -83.76469791  
 -83.70842812 -83.70443295 -83.77401878 -83.70576865 -83.70829848  
 -83.7402872 -83.72566358 -83.78749114 -83.70850876 -83.7083204  
 -83.73783591 -83.7108268 -83.72757097 -83.74076237 -83.70408571  
 -83.73922975 -83.73098727 -83.70981359 -83.69573878 -83.74073716  
 -83.7245562 -83.7087948 -83.70778423 -83.70674734 -83.74066548  
 -83.70896405 -83.70907764 -83.69904545 -83.69365298 -83.70137541  
 -83.71668501 -83.70121176 -83.73999245 -83.71205444 -83.70872736  
 -83.73467785 -83.7033315 -83.71039698 -83.73896343 -83.7280299  
 -83.69169391 -83.70896251 -83.71053403 -83.7499022 -83.68976252  
 -83.717482 -83.70656154 -83.73992399 -83.70874596 -83.71124615  
 -83.70171165 -83.71125195 -83.71140096 -83.71983465 -83.72459138  
 -83.70823236 -83.70913343 -83.70011476 -83.72862742 -83.70688846  
 -83.76038893 -83.73595226 -83.73675852 -83.70692642 -83.76748768  
 -83.70711349 -83.77029101 -83.71838628 -83.7210656 -83.71139881  
 -83.74959766 -83.7013677 -83.68041152 -83.72653887 -83.7051108  
 -83.71348107 -83.72682749 -83.7079398 -83.7041128 -83.73496326  
 -83.72363691 -83.75045406 -83.71028987 -83.72656962 -83.71430372  
 -83.72810436 -83.73001532 -83.71180424 -83.71559609 -83.70490258  
 -83.73049635 -83.74014993 -83.70899466 -83.77971883 -83.70269969  
 -83.68224786 -83.77496899 -83.73160443 -83.73661037 -83.74548205  
 -83.74352254 -83.77075352 -83.70345609 -83.71041183 -83.74033628  
 -83.72884546 -83.71661262 -83.75378006 -83.74003869 -83.68788703  
 -83.72557006 -83.74760594 -83.79884893 -83.73307215 -83.7400481  
 -83.73003061 -83.73985728 -83.74632956 -83.70754894 -83.68557278  
 -83.69323908 -83.71436877 -83.70987452 -83.73114241 -83.72234435  
 -83.73132261 -83.77820982 -83.68721268 -83.73040199 -83.6899288  
 -83.69868938 -83.68030754 -83.740003 -83.69026613 -83.74176329  
 -83.73110896 -83.73983245 -83.75707564 -83.73260906 -83.74953168  
 -83.71441032 -83.68987683 -83.73958097 -83.69861022 -83.79788086  
 -83.72833427 -83.73983881 -83.7354877 -83.69155863 -83.69082721  
 -83.73068244 -83.74733812 -83.68185742 -83.6989775 -83.72737957  
 -83.70171831 -83.6823783 -83.79244549 -83.68404061 -83.76272821  
 -83.68158913 -83.75678044 -83.69479081]  
Longitude[deg]  
-83.694791 1  
-83.792628 1  
-83.719317 1  
-83.725747 1  
-83.725484 1  
 ..  
-83.697277 1  
-83.725189 1  
-83.701236 1  
-83.696121 1  
-83.705607 1  
Name: count, Length: 713, dtype: int64  
Unique values in column 'Air Conditioning Power[Watts]':  
[ nan 0.00000000e+00 7.61245675e-01 2.00872136e+03  
 1.63184822e+01 9.52785146e+01 8.40743735e+00 4.81998530e+00  
 1.23393316e+01 2.44071856e+02 1.85579456e+03 3.89528796e+01  
 4.05347594e+01 2.55272727e+01 2.32584917e+01 1.31511003e+02  
 5.20553936e+01 8.16062176e+01 7.48919827e+01 5.02005731e+01  
 3.17057705e-01 3.93291405e+01 2.46069508e+01 6.07616975e+01  
 1.24148760e+02 5.49573561e+01 8.57222532e+01 9.78629579e+00  
 9.00476190e+01 5.66114502e+01 7.30220493e+00 5.87173913e+02  
 1.63413827e+01 9.56250000e+01 8.00970874e+00 1.86963979e+01  
 1.99204244e+02 1.29632792e+02 1.14680851e+02 2.22717149e+00  
 2.59899610e+01 1.18095631e+02 9.35969868e+01 5.19176598e+01  
 4.65562249e+02 1.52349398e+02 9.76857143e+02 3.29448743e+02  
 1.50000000e+02 6.08314088e+02 6.49795082e+02 3.15129376e+02  
 2.57321131e+02 1.05000000e+03 4.80000000e+02 3.95061728e+00  
 1.24168439e+02 3.70610687e+02 8.28111588e+02 2.84507042e+02  
 7.61000000e+02 1.12252125e+02 3.55162703e+02 2.17677903e+02  
 9.41260745e+01 2.05937136e+02 1.05252890e+03 1.69552239e+01  
 7.52266420e+01 5.50387597e+01 9.42528736e+01 3.21562395e+02  
 9.12408759e+00 6.58203125e+01 9.35404742e+01 1.55840979e+02  
 1.16725352e+02 5.42452830e+01 3.22493225e+01 8.77434077e+02  
 2.43117536e+01 1.79253112e+01 4.71218206e+01 7.87720848e+02  
 3.18718663e+02 2.53405995e+01 7.19280443e+02 3.07692308e+02  
 3.80765456e+01 2.43445693e+01 4.01222754e+02 9.04306220e+02  
 5.17845510e+02 5.18176471e+02 4.47783251e+02 3.48365042e+02  
 4.31927448e+02 3.85466557e+01 1.07282303e+03 2.77796053e+02  
 2.76696970e+02 1.10058252e+03 1.14879650e+03 1.22415317e+03  
 5.39383562e+00 4.00000000e+02 7.28443449e+02 1.05184147e+03  
 1.09933036e+02 2.87425150e+00 3.30708092e+02 4.35107914e+01  
 1.06847826e+03 1.05912763e+03 1.14527244e+03 2.55234591e+02  
 4.58636364e+02 8.97959184e+02 9.00000000e+02 4.70358306e+01  
 1.00812303e+03 5.28597285e+02 1.07926421e+03 7.83061224e+02  
 7.99485597e+02 1.11091463e+03 8.26633987e+02 1.09780362e+03  
 3.57081545e+01 3.48140900e+02 7.90361446e+01 1.78380952e+02  
 2.82629559e+02 4.06828035e+02 1.85946276e+03 8.00000000e+00  
 2.15506958e+01 1.46206897e+02 3.91097467e+01 9.46190803e+02  
 3.50000000e+02 7.85498489e+00 6.84615385e+01 1.08900524e+01  
 8.92200557e+02 2.45768025e+01 1.10723062e+03 5.22689076e+01  
 5.11630695e+02 1.20718563e+01 9.48760331e+01 4.63556851e+02  
 7.37419355e+02 7.63317073e+02 1.22099448e+02 9.55882353e+00  
 4.51991614e+01 4.23778920e+02 7.00000000e+02 4.25531915e+00  
 5.97195358e+02 1.18384146e+03 1.17165017e+03 2.89729730e+02]  
Air Conditioning Power[Watts]  
0.000000 366  
400.000000 2  
0.761246 1  
16.318482 1  
95.278515 1  
 ...   
4.255319 1  
597.195358 1  
1183.841463 1  
1171.650165 1  
289.729730 1  
Name: count, Length: 163, dtype: int64  
Unique values in column 'Heater Power[Watts]':  
[ nan 182.8125 0. 489.45783133 750.  
 218.36577869 134.77537438 349.13793103 401.51515152 220.41984733  
 404.50643777 795. 1390.33166458 911.53295129 713.87283237  
 2096.89922481 54.59770115 83.1381733 1075.84635417 693.66197183  
 763.56132075 193.96551724 533.12720848 564.90250696 2349.84053863  
 1628.0653951 755.53505535 1293.07116105 711.72248804 115.16984644  
 292.94117647 349.57776214 1083.91608392 307.56578947 108.18181818  
 469.41747573 176.1487965 917.89396171 2579.41777324 3014.83420593  
 275.66964286 250. 745.65217391 220.11308562 90.90909091  
 1071.42857143 500. 1302.44479495 101.58371041 1234.65171192  
 3086.63366337 866.83673469 737.65432099 897.25609756 886.92810458  
 877.26098191 203.5765896 838.36650652 3175.30487805 369.31290622  
 554.58412098 3545.28236915 151.07913669 219.75609756 156.81233933  
 1000. 554.64216634 1161.58536585 1115.01650165]  
Heater Power[Watts]  
0.000000 25  
250.000000 3  
750.000000 2  
182.812500 1  
489.457831 1  
 ..  
156.812339 1  
1000.000000 1  
554.642166 1  
1161.585366 1  
1115.016502 1  
Name: count, Length: 68, dtype: int64  
Unique values in column 'Vehicle Speed[km/h]':  
[ 65.29861111 41.42941242 51.80782313 57.99431818 69.11267202  
 40.86466346 67.65190843 80.0574409 39.62506404 52.53676456  
 44.01010708 101.02535408 54.70499067 42.2993271 45.643534  
 32.23121995 43.03157723 21.44891827 54.74823322 37.95580652  
 44.87446844 34.76497459 75.42098472 48.72954408 40.31028664  
 39.79586694 84.78210426 40.17786145 37.50217725 52.38291318  
 39.22908716 46.98022026 33.61496804 46.42862388 47.52056292  
 53.43238786 45.17990729 41.93694982 49.17762937 31.42963371  
 32.06267857 42.18111208 35.92444552 38.76464976 32.90211925  
 43.36172125 24.73029279 56.66242122 43.88892772 77.5641483  
 26.13421296 41.915625 40.71830986 56.14361979 46.85515106  
 44.42310522 40.11094058 46.25475164 28.92735746 25.70928081  
 46.08045664 42.48468828 74.13190737 30.72388549 64.34197092  
 18.05621508 37.05509742 41.99257991 36.66341808 45.54867661  
 65.00378024 48.16351513 43.49981655 49.1031746 32.3418509  
 34.90525314 36.04573752 41.19079603 41.93461503 50.09455552  
 46.59761376 21.90002991 34.61319033 48.70837048 39.83763923  
 29.4494405 39.61196977 83.17097387 86.04003405 28.2272634  
 34.74702091 33.44701951 35.82292816 19.96722561 18.53769697  
 37.41024016 31.89139803 67.97251406 40.8373494 32.21184585  
 55.84051067 15.17016807 32.87872432 32.30424763 33.32378787  
 30.61749331 37.77878676 41.86628845 44.29038443 33.71354167  
 21.9848521 40.38647201 42.11035547 40.22079813 29.54329044  
 34.25935059 33.29164514 32.05590753 22.65295493 23.25638507  
 29.42705989 27.9430776 27.846431 27.72562029 35.65039567  
 38.63815214 34.7787238 26.00800449 26.39455275 45.23936632  
 38.81057269 29.76169305 37.44062054 45.17492884 35.52584496  
 36.96068403 50.0139523 45.18004052 28.37114396 35.74339079  
 57.88322996 38.3888851 43.94560918 39.44244442 39.99183437  
 35.42769282 34.48172923 41.0936696 45.86533289 31.17307918  
 45.29593363 40.76183874 32.37513031 34.95485858 32.50708082  
 31.97550032 42.84915304 42.66831282 45.35620635 38.08846416  
 34.63677573 46.24073998 40.53177966 34.4036287 28.23242555  
 43.40205841 46.57986365 38.33527692 50.7091763 45.3359746  
 46.77216717 38.80373775 31.9603988 38.2142503 39.85616865  
 38.17790354 30.94187986 30.85708117 36.51133457 36.08142846  
 25.45249122 56.86521641 38.56176297 43.36458333 54.74894104  
 42.83868436 45.75788597 51.25215841 41.85030977 44.36453459  
 46.29894052 49.68256048 36.94628251 48.80609756 43.73315386  
 48.07048611 39.06659226 42.05384291 68.63097994 61.10588431  
 56.59685617 65.38540285 61.63919483 40.9723424 48.88802083  
 53.36122159 40.72042872 62.0864547 34.45178167 50.55454009  
 72.18875839 54.20297965 79.34766695 62.23580918 67.80899701  
 50.69966133 42.92180333 62.94433594 66.03291561 47.85709064  
 56.54186809 70.25849865 59.36247417 76.34099835 59.15512725  
 57.225804 74.09182007 55.09222342 63.85271288 43.61591734  
 25.86999955 52.82094595 49.31379467 44.18748364 42.13190608  
 39.51890538 47.2889334 22.87550491 51.34683544 36.10120713  
 43.5627902 44.07578558 47.36330935 41.82279441 42.68716578  
 32.50127747 32.26949016 39.58351751 38.99331713 55.92334241  
 42.52148813 27.67738744 26.43505674 51.01125541 34.3845711  
 69.61981401 25.56229423 33.80466472 30.28747941 22.15475376  
 44.91859597 43.02266289 34.09583437 38.28925364 49.80690827  
 19.13203614 55.57326892 38.32864545 36.29454023 45.38686346  
 45.42054795 36.124834 29.98535766 32.69789864 46.00507356  
 34.78909774 39.62378223 36.95306197 40.99637562 37.32276474  
 43.3120788 34.42611821 40.85424438 42.95620307 18.48948995  
 31.06288788 48.26499904 46.0316092 49.11917199 14.97712284  
 38.20600581 41.09148988 43.82720307 44.55631612 48.64308943  
 18.12750371 35.78784337 38.32468335 43.34399431 34.66280564  
 42.33100948 29.80378049 76.71948209 70.22293871 46.52563122  
 37.46858485 38.17424705 30.92647777 39.0928382 34.93432142  
 27.54474708 33.60711462 39.19153159 42.8778433 46.20174898  
 51.03987268 31.72065095 28.465063 37.37709392 40.66920943  
 33.49892646 27.87457162 38.93213511 29.07657839 49.64442636  
 44.98254364 44.07120084 36.29753915 37.59790875 20.84727216  
 37.46350737 36.26688233 34.92738708 37.71706454 39.27962085  
 45.57771098 53.76623576 26.42895763 40.26523865 48.21473826  
 60.47475379 39.24790221 39.47095179 40.5246897 24.3234655  
 48.40909091 29.92468619 50.70931294 46.55037465 47.29248708  
 42.59948762 35.87872271 33.18154138 40.24329279 37.48937583  
 45.62325381 36.01975295 34.10022246 38.59989643 55.34330247  
 47.81995988 34.39217871 34.48692103 41.05935697 39.87452107  
 27.5387761 51.1262733 46.86835106 46.49695122 40.29072923  
 44.05317588 60.56915119 53.04015605 38.94555874 54.83840781  
 51.8888278 42.82989126 39.95994647 46.7079946 47.18882329  
 44.40649108 43.64916064 3.49054212 51.34059877 53.  
 45.61935164 45.76562394 47.20535286 29.56767641 69.30096419  
 31.81282496 37.51559809 23.54857143 31.1605953 31.96963351  
 46.84729815 23.56215128 31.68550353 25.81934379 16.74364896  
 43.12189596 19.26589606 44.17226693 27.66141375 41.36350493  
 31.38204302 22.15922956 18.65333868 46.22749957 39.85946891  
 31.78957548 42.52867092 33.40930954 43.59554674 30.84224201  
 40.31780589 25.43983478 37.75585744 36.69265537 39.16508229  
 35.25021194 43.38540333 46.78812874 47.26038516 47.61338517  
 33.262112 37.50037527 35.92865158 12.98654967 42.19830671  
 41.52635896 23.85657015 24.24825289 40.0232388 25.38388917  
 29.74893909 27.1285093 42.50106452 31.01491389 39.12405984  
 49.50675515 42.66319051 35.55007676 30.45284411 33.72005572  
 39.50434198 34.00935024 42.65242611 27.50951753 34.61907498  
 22.26607472 36.4236488 66.74978443 33.61699609 40.11065211  
 47.51554928 26.36556278 40.98351182 20.40884385 35.41535155  
 45.60867196 37.68225701 40.67636064 30.72509693 27.07272247  
 29.58139535 31.65313653 43.55365473 38.01667644 29.28036455  
 41.86126733 48.49123089 88.81640683 30.87575067 48.79563419  
 16.15047681 31.96295784 30.50040558 39.5209295 37.89670796  
 40.61525962 12.18586792 19.37874621 38.66346763 43.05190762  
 39.34754639 34.22910181 31.44163819 22.92509312 42.31972792  
 39.35507062 57.78833523 33.36173961 32.32528163 41.50075755  
 36.45755515 41.60045537 39.60508355 48.29627717 29.43707101  
 39.58992865 39.68185492 23.62513713 31.76799266 40.87895545  
 42.62297465 51.59813088 41.93515389 40.01754854 41.63875351  
 50.41912005 30.34182457 45.2918238 29.67794251 36.47698238  
 39.43554953 43.50627288 47.81059929 23.99731903 48.97108092  
 33.89833261 35.29907056 41.10706491 37.88774789 27.65004032  
 32.09926185 57.82982752 31.69486844 38.52402377 51.92610555  
 45.11870101 52.80319361 36.60128102 24.03584767 40.45847801  
 45.51314924 28.35791543 42.91208531 46.78066381 41.98472876  
 42.13301538 40.67496112 35.0194008 41.02827719 26.44165161  
 42.8479635 42.31366079 40.20032051 35.76560171 31.31272657  
 42.52985146 5.59891138 49.75183094 42.17597417 35.8712804  
 40.11504647 34.28507611 47.1844265 50.64431838 51.04596109  
 30.61764783 38.56521739 32.74804009 43.10020477 40.45387716  
 45.58633569 58.06707883 49.96546231 48.77627465 29.88241601  
 42.72674002 45.82092576 48.06601323 29.18201854 41.59673828  
 47.99762211 44.29941861 43.21476742 48.94968513 51.98693908  
 40.90643082 48.16350168 25.1987074 41.89797234 42.66991318  
 43.93647648 50.76009223 46.13935653 33.94414604 42.98654189  
 49.02624608 45.20063474 46.50839707 51.33662043 44.64723977  
 44.85601058 39.32967033 38.31195547 43.55527318 38.04953194  
 57.92473405 49.46552917 39.01067258 42.89159787 43.84229378  
 30.41697373 43.52297728 47.180466 29.02459788 47.29619623  
 52.62043811 36.47004481 44.87088541 44.16411262 49.78826206  
 53.44019514 29.16451864 50.94109408 45.21475035 30.64662343  
 26.99407545 46.99847121 42.66521473 46.78356441 19.09691677  
 51.97670533 43.05063035 44.18609807 61.29270903 56.73104895  
 42.51492347 25.55649109 26.14321528 36.74622318 40.64999913  
 33.45108501 33.7284359 39.33517911 50.86997688 47.90150445  
 46.3027823 34.48548425 53.04437696 42.78112518 47.08489571  
 64.25016848 67.57028915 40.9401443 46.26126454 56.25490795  
 54.61997169 41.25897866 51.12978416 49.97254705 40.0129117  
 43.98407309 43.30070215 41.29882533 39.0819612 43.50787007  
 46.58815511 46.05356403 76.91285166 45.17099907 35.52449797  
 44.75478201 45.23636419 54.88884493 42.56776109 52.97312389  
 42.26488162 47.30612346 49.71215343 44.73288101 59.80466073  
 50.08383463 43.76704311 48.80172785 50.60673906 65.42049982  
 51.57100288 45.86853839 44.17478219 54.90749868 64.90858792  
 45.48150336 46.35485082 44.18545116 45.50893588 36.46665285  
 65.35596032 60.65778715 38.5067149 38.88361199 66.40044995  
 42.64482836 48.24326502 58.25115741 65.12189291 81.18149407  
 53.847047 39.09767576 79.24437532 49.40645919 51.36078198  
 55.02486344 59.11695213 65.89523022 66.77167009 52.89353713  
 83.19398038 59.7251502 88.79979448]  
Vehicle Speed[km/h]  
88.799794 1  
65.298611 1  
41.429412 1  
51.807823 1  
57.994318 1  
 ..  
42.299327 1  
54.704991 1  
101.025354 1  
44.010107 1  
52.536765 1  
Name: count, Length: 713, dtype: int64  
Unique values in column 'Distance[km]':  
[8.18788433e+00 2.67876585e+00 5.43478806e-01 5.30422585e-01  
 5.85098222e-01 3.82667785e-01 4.12244057e+00 3.80198142e+01  
 6.15052579e-01 2.80944406e+00 8.11561350e-01 7.11232572e+00  
 4.79798640e-01 1.26134592e+00 2.83596208e+00 2.61398721e-01  
 1.20428164e+00 3.43792624e-01 1.03981873e+00 7.06038265e-01  
 1.91426580e+00 3.72146588e+00 5.96176213e+00 1.63550753e+00  
 3.59176026e+00 4.00699373e-01 5.19323249e+01 9.29620307e+00  
 3.99696305e-01 3.32467225e+00 2.22085062e+00 2.57462850e+00  
 5.81702267e-01 4.36388480e+00 4.05544241e+00 6.16561461e-01  
 2.13522098e+00 8.77366202e-01 1.33996544e+00 5.07599698e+00  
 1.89431291e+00 3.19465882e+00 7.91587093e+00 2.16024680e+00  
 5.85807130e-01 5.28546122e-01 5.05235325e-01 5.83354356e-01  
 5.22164110e+00 3.18653864e+00 7.44120448e+00 5.97739657e+00  
 3.69764020e-01 5.18368377e-01 1.91920854e+00 4.02924932e+00  
 6.80834265e+00 4.16697279e+00 4.35181854e+00 1.64902446e+01  
 2.12071800e+00 9.93660847e-01 3.60760144e+01 4.52185236e-01  
 9.67565818e-01 2.30135703e-01 1.17334512e+00 8.97688016e+00  
 2.13470964e+00 2.08616187e+00 5.76704103e+00 8.07048217e+00  
 2.23986715e+00 8.15328348e-01 3.08607178e+00 7.87936386e+00  
 7.56650060e+00 1.85628277e+01 2.70745476e+00 3.37229876e+00  
 1.64398200e+01 1.10868694e+01 6.05415348e+00 2.78831547e+00  
 1.29755825e+00 2.89980959e+00 1.13311920e+00 7.02455879e+00  
 7.55148220e+00 1.15424809e+00 6.37022434e-01 7.41743257e+00  
 6.16318861e+00 1.21910635e-01 2.82430182e+00 3.85850461e+00  
 2.19429679e+00 6.02786685e+00 8.38239838e-01 9.99137399e+00  
 4.47379010e+00 4.50766551e-02 1.23539198e+01 1.26929152e+01  
 2.97041079e+00 2.81788586e+00 3.26196520e+00 2.37777925e+00  
 3.24283894e+00 2.09281436e-01 3.41129044e+00 4.99852820e+00  
 5.40557512e+00 3.19736347e+00 1.19275176e+00 2.25350566e+00  
 1.32080099e+00 1.14981726e+00 1.26332968e+00 1.17376480e+00  
 3.28709860e+00 3.59642362e+00 3.39030981e+00 9.37825738e-01  
 1.43911635e+00 2.84519411e+00 6.24387270e-01 1.31232767e+00  
 3.41046388e+00 7.68028607e-01 7.79359100e-01 5.87866352e+00  
 3.13008793e+00 1.87481498e+00 3.59228830e+00 3.05482513e+00  
 4.09273574e+00 2.19708262e+00 7.06876577e+00 2.06953484e+00  
 6.37261275e-01 2.94696641e+00 1.01589221e+00 2.97475338e+00  
 8.36788377e+00 1.64709263e+00 1.83757762e+00 2.12177759e+00  
 1.42409203e+00 4.26836779e+00 2.79235693e+00 3.02746581e+00  
 2.36373085e+00 3.00526424e+00 2.51166936e+00 2.67542427e+00  
 8.28616230e-01 8.33197728e-01 2.53550089e+00 2.69057427e+00  
 2.73689147e+00 3.62373662e+00 7.33261196e-01 3.47910476e+00  
 2.57511061e+00 5.96209642e+00 3.79914272e+00 4.93215951e+00  
 7.42155525e-01 5.01287490e+00 9.54803029e-01 9.89908981e+00  
 1.24082572e+01 1.54492533e+01 6.85032408e+00 8.21821625e-01  
 8.43897080e+00 1.87072637e+00 7.17085825e+00 3.82792517e+00  
 3.73074134e+00 4.62616445e+00 1.04511435e+00 3.38497414e+00  
 3.80908420e+00 2.15681676e+00 1.69636244e+00 2.12916841e+00  
 7.88622578e-01 1.01181677e+01 8.31248025e-01 1.51824098e+00  
 1.60643406e+00 7.90118166e-01 2.47114432e+00 2.47062569e+00  
 8.89209532e-01 8.48514568e-01 1.01753841e+00 1.00752788e+00  
 1.71810454e+00 2.64857050e+00 4.51618366e+00 6.38005586e+00  
 6.04931032e-01 1.19641642e+00 9.13652448e-01 1.74186724e+00  
 1.07628670e+00 1.06195901e+00 1.02817525e+00 1.17535031e+00  
 4.70964373e+00 1.13335350e+00 5.52934347e+00 1.05603982e+00  
 8.67739643e-01 1.01851360e+00 1.20637108e+00 7.87225897e-01  
 8.86528974e-01 2.84214018e+00 4.03190832e-01 1.00044179e+01  
 8.92327759e-01 1.13025384e+00 2.37069530e+00 9.96250723e-01  
 6.76900603e-01 4.83936778e-01 2.61166470e-01 1.21567586e+00  
 7.80887477e-01 9.83153779e-01 3.92110850e+00 5.00505451e-01  
 5.66881887e+00 4.90651984e+00 4.63748302e+00 3.75325042e-01  
 3.91142326e+00 3.48823426e+00 5.35695078e+00 4.24213431e+00  
 6.75021595e+00 4.58213203e+00 1.21477142e+00 3.01050370e+00  
 4.84418530e+00 2.43763649e+00 5.10689477e+00 1.47179006e+00  
 1.27841997e+00 4.80302350e+00 5.09462090e+00 3.77924197e+00  
 1.30699074e+01 1.36441772e+01 1.32681469e+01 2.30980086e-01  
 4.26238733e+00 8.80044440e+00 4.76259615e+00 9.06194882e+00  
 7.13785093e+00 4.70072328e+00 2.53375546e+00 3.83198875e+00  
 3.84845855e+00 4.81710857e+00 4.66471788e+00 4.20944109e+00  
 1.17193346e+01 1.13037092e+01 3.41958052e+00 3.57043819e+00  
 1.01504016e+01 1.47834978e+01 1.00418369e+01 9.14403907e+00  
 1.24348772e+01 1.00215505e+01 2.67504193e+00 3.23371388e+00  
 5.32627547e+00 1.27375589e+01 6.34832945e-01 2.75275067e+00  
 3.34770184e+00 2.44667046e+00 1.01079751e+01 3.65009652e+00  
 8.03096168e+00 4.42212957e+00 3.71550165e+00 2.13610418e+00  
 3.53966155e+00 8.76884448e+00 8.22246782e+00 7.10111487e+00  
 1.13800143e+01 1.13730296e+01 8.38926398e-01 5.58972742e+00  
 4.58484921e+00 3.18324780e+00 1.14074650e+00 1.17073548e+01  
 3.32066070e+00 9.71592461e+00 6.75834786e+00 1.23013136e+01  
 8.68354873e+00 3.51802099e+00 8.81168547e-01 1.63147677e+00  
 3.14012365e+00 2.47655715e+00 4.80306756e-01 3.20662136e+00  
 1.09655899e+01 3.07567602e+00 9.82770962e+00 9.53130161e-01  
 2.98568029e+00 3.60274058e+00 2.55899349e+00 4.46016827e+00  
 9.01051496e+00 1.87478454e-01 4.52236928e+00 1.06647543e+01  
 3.67598499e+00 4.54362479e+00 3.53288088e+00 3.13441617e+00  
 2.14744608e+00 6.34208456e-01 1.04008916e+00 6.95589207e-01  
 3.54662117e+00 2.80847802e+00 3.28550148e+00 3.63305138e+00  
 5.55506229e-01 1.31213000e+00 2.92374967e+00 2.94603731e+00  
 7.49020395e+00 3.00438530e+00 4.71693621e+00 6.71338123e-01  
 5.73785673e-01 6.51928373e+00 3.27264844e+00 3.65581629e+00  
 3.17533911e+00 5.55708381e-01 2.95245752e+00 1.99368938e+00  
 1.27424985e+01 4.53811546e-01 3.45655771e+00 3.08654695e+00  
 3.35164969e+00 7.32811897e-01 5.08663224e+00 3.20918403e+00  
 4.11669433e+00 7.57423215e-01 3.74680962e+00 4.56193150e+00  
 1.76105656e+00 1.11095575e+00 5.57468059e-01 4.95306582e-01  
 9.02818057e-01 5.08726959e-01 3.83900315e+00 1.07108320e+00  
 3.25452283e+00 3.01238710e+00 2.12290052e-01 6.06407553e-01  
 4.85853803e+00 3.44415464e+00 1.52985501e+00 2.47666363e+00  
 1.08996652e+00 7.73304012e+00 2.63977725e+00 1.01601018e+00  
 1.63058254e-01 8.96625202e-01 2.25919948e+00 2.40292791e+00  
 6.97959363e+00 5.68204728e-01 1.74693409e-01 3.91454593e+00  
 4.00606686e+00 7.07219877e+00 1.97309744e+00 1.66102587e+00  
 2.60138127e+00 1.27699821e+00 7.24157614e+00 1.84179301e+00  
 8.22159871e-01 4.74433369e+00 4.25335917e+00 2.94826418e+00  
 4.60863758e-01 2.70527173e+00 3.20840956e-01 3.97337889e+00  
 1.36959559e+00 1.22788828e+00 3.17444862e+00 2.95913894e+00  
 4.17394266e+00 2.22765593e+00 6.86828789e-01 9.95023613e-01  
 1.00985473e+00 9.21343636e-01 3.00509236e+00 4.80872220e+00  
 2.47652160e-01 1.33218895e+00 3.51956715e+00 5.33205506e+00  
 3.45451637e+00 2.39903115e+00 3.54095519e+00 1.80862110e+00  
 1.65950725e+00 1.93137829e+00 2.83225271e+00 1.42314096e+00  
 2.15404496e+00 3.92313530e+00 4.90033813e-01 9.61780629e-01  
 3.25095753e+00 6.46480431e+00 6.90527365e+00 3.57570452e+00  
 1.78680571e+00 3.05030043e+00 2.84394719e+00 2.11353958e+00  
 3.68479047e+00 3.79848496e+00 4.15406714e+00 2.63939821e+00  
 6.26353355e-01 4.00002931e+00 5.54172296e-01 2.01986019e+00  
 2.45965284e+00 3.38459984e+00 6.27620608e+00 9.51630730e+00  
 1.40103475e+00 6.94690999e-01 1.60196407e+00 1.71187516e+00  
 3.56705095e+00 3.13541971e+00 3.75267865e+00 3.98669769e+00  
 6.05746262e+00 5.56762653e+00 2.49942098e+00 1.37747864e+00  
 1.89705101e+00 1.06227807e+00 1.01550463e+01 2.95852790e+00  
 2.47852273e+00 2.41035825e+00 9.20744710e-01 1.57903825e+00  
 3.94780684e+00 1.73955896e+00 3.65534860e+00 3.43437472e+00  
 1.43333652e+00 5.88257679e+00 6.21054577e-01 2.30842266e+00  
 6.07791045e+00 2.63657544e+00 3.79661571e+00 2.04535407e+00  
 2.89187274e+00 9.02626572e+00 6.27579935e+00 1.10310754e+01  
 2.81131969e+00 3.61596675e+00 3.91095918e-01 5.83536371e+00  
 6.23928594e+00 3.05417859e+00 2.19053679e+00 2.62992174e+00  
 1.57391101e+00 5.85855728e+00 5.61124138e+00 1.70480001e+00  
 2.62454792e+00 7.63127657e-01 1.11873797e+01 2.32960844e+00  
 2.03457107e+00 2.93862431e+00 8.17387697e-01 3.26281689e+00  
 7.58371137e-01 1.54281477e+00 6.23159423e+00 2.71185514e+00  
 2.81099093e+00 1.09610284e+00 4.44270185e+00 1.12452102e+00  
 7.03201352e+00 1.14871669e+00 5.17556924e+00 6.83900371e-01  
 4.19435457e+00 1.22275822e+00 2.92816195e+00 2.27871750e+00  
 3.25083517e+00 3.19663819e+00 3.33174796e+00 5.09837137e-01  
 6.63104302e-01 2.25241731e+00 1.04766603e+00 6.44269184e+00  
 2.02796162e-01 1.99077594e+00 2.54556702e+00 4.26565928e+00  
 4.76763164e+00 3.79810367e-01 3.17326151e+00 1.98648247e-01  
 4.10950794e-01 2.61394788e+00 1.57787544e+00 1.63209240e+00  
 3.94376957e+00 2.83424588e+00 2.32488075e+00 3.44612855e+00  
 1.85376001e+00 4.08548449e+00 9.87875138e-01 1.53272622e+00  
 2.74911988e+00 3.52482917e-01 3.46978512e+00 5.17384091e+00  
 2.36744578e+00 3.25588187e+00 2.67176386e+00 2.83341800e+00  
 2.39889195e+00 8.80207850e-01 4.91884285e+00 5.74736796e-01  
 2.86534626e+00 2.49634994e+00 5.11491958e+00 4.60788017e+00  
 3.06894913e+00 4.79606869e+00 6.44410314e-01 3.01330202e+00  
 3.59352853e+00 2.08908319e+00 2.62140880e+00 1.12686370e+00  
 2.11499388e+00 4.55850622e+00 2.26902967e+00 2.24516363e+00  
 2.55358532e+00 1.50046440e+00 3.12266355e+00 5.24552566e+00  
 2.13254060e+00 2.24728094e+00 1.42214632e+00 3.20279339e+00  
 5.61419699e+00 4.33823406e+00 2.22481325e+00 3.77886078e+00  
 4.05885529e+00 3.37660108e+00 2.21527474e+00 1.17806205e+00  
 3.56432860e+00 2.39936149e+00 9.09518562e-01 1.60154583e+00  
 3.95130715e+00 2.41452711e+00 3.95446826e+00 3.22320022e+00  
 5.87025461e-01 1.73240056e+00 1.68798629e+00 7.06594974e+00  
 2.73727289e-01 2.89442927e+00 6.42136699e-01 3.78363548e+00  
 1.88489488e-01 2.40268096e+00 3.72684874e+00 2.52531735e+00  
 6.39123883e-01 3.28455026e+00 3.23876828e+00 1.04595868e+00  
 5.87769895e-01 5.22509168e+00 6.80105534e-01 4.41122613e+00  
 3.41344571e+00 9.77735278e-01 1.76440963e+00 2.41002827e+00  
 4.11467789e+00 3.81721666e+00 3.47051033e+00 2.13057538e+00  
 2.14933678e+00 4.47263053e+00 4.45927690e+00 1.55248148e+00  
 6.86280914e+00 2.70517892e+00 4.73919289e+00 8.77449269e-01  
 2.79787363e+00 3.03838751e+00 2.72365314e+00 1.84846247e+00  
 4.94833330e-01 1.42354440e+00 3.16629853e+00 4.72854689e-01  
 3.50413675e+00 2.27128858e+00 2.75771584e+00 3.61683757e+00  
 3.24065920e+00 3.90305212e+00 2.51267508e+00 2.95530609e+00  
 2.46560958e+00 2.49822474e+00 2.93932066e+00 3.88998185e+00  
 1.75775493e+00 3.91508986e+00 1.78252105e+00 3.88656654e+00  
 2.07129592e+00 9.28286198e-01 4.42863787e+00 7.77025055e+00  
 2.54167639e+00 6.70429597e-01 3.31413651e+00 2.06876370e+00  
 6.29382562e-01 4.26535244e+00 2.63870175e+00 1.52989937e+00  
 2.46190789e+00 2.23652918e+00 1.57413431e+00 2.06259035e+00  
 3.13430949e+00 7.25173772e-01 3.73229307e+00 4.33399481e+00  
 2.69612811e+00 6.00936980e-01 7.50840985e+00 8.07756888e-01  
 4.00438186e+00 2.76032959e+00 2.17575916e+00 5.03298428e-01  
 2.75392217e+00 6.07762483e-01 1.78524026e+00 4.65986403e+00  
 4.07160245e+00 1.44945081e+00 1.94224766e+00 1.02810871e+00  
 8.15004879e+00]  
Distance[km]  
8.150049 1  
8.187884 1  
2.678766 1  
0.543479 1  
0.530423 1  
 ..  
1.261346 1  
0.479799 1  
7.112326 1  
0.811561 1  
2.809444 1  
Name: count, Length: 713, dtype: int64  
Unique values in column 'Engine RPM[RPM]':  
[2105.89569161 498.72362685 2074.41496599 2231.63636364 1877.91743119  
 1849.83846154 1679.01337296 1662.74450812 1628.34631148 676.68165785  
 1612.26573427 1714.48960499 1525.67164179 1514.60948905 1536.28762542  
 1353.67307692 1365.92408377 1218.82478632 1453.54063604 1573.1320132  
 1065.38764548 611.6778626 1258.4954583 1207.50483092 977.20233266  
 1226.70967742 1115.54345794 742.36566265 1326.82786885 1197.47551546  
 847.06396867 1282.57251908 1019.59659091 1090.44920525 1175.96501129  
 1017.88654354 1053.04603581 1173.45422535 1158.20698925 1061.9707401  
 1183.06666667 966.97444832 589.44468784 1066.45183175 1048.09770115  
 888.66134185 692.17117117 982.98319328 936.52502662 877.10115607  
 852.4582716 927.8900369 851.84507042 938.91666667 710.77319588  
 932.20941558 746.90177515 1096.56454389 759.57309942 733.27556512  
 680.78682171 1401.44611529 608.37266418 644.39160839 1399.96039604  
 514.77653631 436.54320988 360.2260274 521.56158192 793.88520179  
 381.31265509 393.25477707 644.56763926 414.37037037 1000.72422464  
 320.84291928 321.40826365 278.48071536 343.49778434 323.2881692  
 305.53081943 277.70967742 265.94498519 110.1814882 334.60643564  
 282.20815138 215.22382671 376.35530381 250.23910256 306.43763676  
 277.60311284 204.21618497 205.31052244 537.7195122 510.5625  
 181.49403341 284.07157895 337.0794176 143.17269076 33.17312602  
 61.48780488 0. 678.26682937 524.15122073 460.94510616  
 596.01593625 667.76411765 550.6038961 594.83689024 446.06614786  
 621.18025223 471.82744496 350.22752809 225.71096654 421.58375315  
 196.40261438 128.45043732 527.54335895 2019.68571429 705.19727891  
 923.03355705 1220.83783784 913.02617801 1111.35582822 833.50690608  
 596.1328125 863.30195293 432.71539658 939.05443038 589.65100671  
 764.61318681 764.82439926 1027.77985612 775.6655574 725.54705882  
 777.54215636 678.29818182 882.32186459 727.80315917 1099.97852761  
 799.72161642 493.55597015 507.8129771 1251.9004329 751.88831318  
 1122.00492005 650.0149021 724.78309038 578.82125206 624.18032787  
 915.15384615 808.04603399 604.95364238 735.63301332 916.11494904  
 644.64280751 1057.93880837 678.22945966 712.31106322 1071.03655565  
 756.82328767 773.62881806 626.43951032 684.58887515 1049.14925373  
 1132.23234201 671.90916905 766.39860653 771.45601318 730.53424223  
 764.50259202 639.93889776 803.94968553 856.53417455 565.97399896  
 729.74762102 1063.19548872 1194.16762452 731.55168662 447.02040396  
 679.9719083 801.98644724 741.5045977 1451.97448662 874.8796748  
 542.72479339 801.3798948 793.92740192 816.44314144 696.58640104  
 761.39905187 775.21845319 1782.4 1414.6 740.42999235  
 715.47856723 588.55994898 566.401319 759.7683466 683.04824179  
 988.37924773 833.23833992 728.05193516 1132.8989048 nan  
 1199.88383838 622.41427463 733.04467354 788.10397554 706.92718447  
 763.60332797 856.96780129 748.96591261 441.72368421 866.91381346  
 851.90024938 1034.96153846 736.20917226 741.87880228 344.64935065  
 680.2834769 711.5430651 727.43167524 866.37046861 853.7718348  
 1283.773238 1213.32989691 476.32738095 856.77245509 900.67826087  
 1036.10788382 811.09412861 858.88195303 641.91748879 624.92430279  
 909.71343874 978.66276151 840.260631 848.23239437 784.04285714  
 909.31596926 663.14893617 488.58878505 841.62301587 661.30876494  
 807.43137255 1250.52 585.06013363 953.6066838 1114.39786856  
 892.40602342 307.93432836 672.53257329 676.11211871 780.24329502  
 587.87155963 898.94304388 956.80585106 877.77743902 783.17082917  
 987.86817326 898.03143418 636.48710602 893.20967742 1174.67368421  
 976.37486457 763.39628483 601.53463588 892.72145615 1101.072  
 3360. 606.328125 1245.5 802.625 527.7630662  
 1210.45247934 356.82322357 284.61281337 1076.81163435 688.08453608  
 857.47640791 543.68952135 317.41542625 398.94233129 397.18148148  
 633.23488603 377.73883162 1088.25718016 260.97290152 347.77881752  
 856.13316261 821.03366337 231.87525816 525.59011164 643.2991453  
 952.35024876 592.78445883 289.80588027 222.19160584 957.89296636  
 543.20272904 829.54280671 406.92475068 528.16685979 414.53017751  
 196.73226545 396.25193798 171.93418259 529.2105594 1598.19830713  
 845.23522013 310.66845878 273.38567223 785.14766355 207.93090211  
 829.27522936 59.33773741 139.08759124 267.00039078 580.19047619  
 355.60400286 258.52625938 429.20471281 319.78538103 355.2202729  
 287.85731857 1044.63347165 302.8365651 818.54306569 375.90041494  
 593.77586207 92.70367208 357.98017348 1119.19714286 410.25043478  
 809.30955121 409.86083499 801.69525547 331.0261233 285.69455104  
 699.73322148 444.88425443 1098.0787949 301.3109405 532.62537764  
 145.28010471 319.69456265 628.06521739 1215.67171717 1059.45323741  
 1090.16199849 496.14551084 594.55799373 108.91896145 722.25272331  
 971.09734513 295.40378343 611.63636364 185.59111111 210.02718447  
 541.05316456 238.20677362 473.06521739 50.84725537 324.93396226  
 233.16923077 6.45696401 133.61389173 118.82666667 850.59661017  
 108.37283237 277.24458874 397.4893617 367.72757475 74.43872549  
 359.78064516 229.12152778 186.14678899 1327.84251969 156.56626506  
 330.05365854 123.52488688]  
Engine RPM[RPM]  
0.000000 238  
2231.636364 1  
1877.917431 1  
1849.838462 1  
1679.013373 1  
 ...   
977.202333 1  
1226.709677 1  
1115.543458 1  
742.365663 1  
2074.414966 1  
Name: count, Length: 381, dtype: int64  
Unique values in column 'OAT[DegC]':  
[-1.86746032e+01 8.86268527e+00 -3.00000000e+00 7.00000000e+00  
 3.61467890e+00 -8.38461538e-01 2.09658247e+00 -5.11294174e+00  
 1.68852459e+00 8.31856261e+00 -6.84440559e+00 6.09615385e+00  
 -1.05000000e+01 4.00000000e+00 1.28494983e+01 -6.50000000e+00  
 2.17277487e+00 -1.60000000e+01 -4.90106007e+00 -5.23267327e+00  
 2.35362578e+00 -6.45203562e+00 1.26878613e+00 5.19323671e-02  
 -3.09875761e+00 -7.50000000e+00 -1.78130841e+01 8.20180723e-01  
 2.50000000e+00 3.26224227e+00 6.79781332e+00 3.50000000e+00  
 -1.00000000e+01 -1.85110574e+01 3.14672686e+00 -8.36411609e-01  
 9.00000000e+00 1.40000000e+01 4.45161290e+00 8.01075731e+00  
 5.43619048e+00 -1.02137050e+01 -5.58180449e+00 6.00000000e+00  
 3.00000000e+00 -2.57188498e+00 -1.50000000e+00 5.95580405e+00  
 -5.87500000e+00 3.49283951e+00 8.60553506e+00 -5.95131730e+00  
 4.25405844e+00 1.25049310e+00 2.09358864e+00 -4.94093567e+00  
 2.83100108e+00 8.25193798e+00 -1.50977444e+01 8.14655633e+00  
 1.25629371e+01 -7.50000000e-01 1.59776536e+00 -7.87962963e+00  
 -6.42768362e+00 -6.90381166e+00 4.25992556e+00 7.50000000e+00  
 -1.14235190e+00 -1.09814815e+01 -4.17770327e+00 3.90623490e+00  
 8.34849975e+00 -7.10439560e+00 -9.84711965e+00 1.26272307e+00  
 7.68237854e+00 -2.48316200e+00 -4.91599661e+00 2.69963702e+00  
 -1.18118812e+01 -1.14039301e+01 4.96389892e-01 2.84191555e+00  
 1.16076923e+01 -9.50000000e+00 -1.01575875e+01 -1.14413295e+01  
 8.00699043e+00 1.00000000e+00 5.91896186e-01 -1.09612172e+01  
 -1.68421053e+00 -1.25148908e+00 -4.00000000e+00 6.89556742e+00  
 1.50609756e+00 6.50000000e+00 -4.22005288e+00 1.26594420e+01  
 -5.12402900e+00 2.49676295e+00 3.22558824e+00 2.47970779e+00  
 6.89710366e+00 -1.00000000e+00 7.80725191e+00 5.35924125e+00  
 -1.74207936e+00 6.40732207e+00 -9.21568627e-01 -2.60156250e+00  
 1.09793388e+01 -2.95011338e+00 -5.00000000e+00 2.57022472e+00  
 -2.00000000e+00 -1.46859296e+00 5.75710594e+00 1.40508982e+01  
 -4.32797732e+00 2.66146789e+00 1.20000000e+01 1.10000000e+01  
 5.37458882e+00 4.65845725e-02 1.13662239e+00 1.02147213e+01  
 6.98881239e+00 4.12153652e+00 -3.50000000e+00 6.29627249e+00  
 7.75970217e+00 -5.00000000e-01 6.90065502e+00 3.65392157e+00  
 7.48100134e+00 -1.42092652e+00 5.50000000e+00 5.80955777e+00  
 5.64943609e+00 4.64880952e+00 5.29351536e+00 3.79702970e+00  
 8.03941606e+00 7.53121853e+00 -2.50000000e+00 4.14719626e+00  
 5.22279793e+00 3.10854583e+00 -2.12281341e-01 -2.15316045e+00  
 8.49605782e+00 4.08474576e+00 4.39692982e-01 -4.85980479e+00  
 8.68734257e+00 4.17597765e-01 6.75197294e+00 -2.65895954e-01  
 7.07217474e+00 1.25000000e+01 8.12072829e+00 -3.07769929e-02  
 5.17251857e+00 7.62018256e+00 5.86614173e+00 -8.97927461e-01  
 6.28538943e+00 8.31773998e+00 2.53900156e+00 -3.27799122e-01  
 4.50000000e+00 -3.98726852e+00 7.70401107e-01 7.99391172e+00  
 -2.43103448e+00 6.71471471e+00 2.36664439e+00 7.10275689e+00  
 9.50000000e+00 5.15939597e+00 -4.84878049e+00 5.40586146e+00  
 1.16606061e+01 1.17459459e+01 1.07712766e+00 2.09638554e+00  
 4.47877984e+00 6.77131239e+00 -1.06390606e+01 4.18815331e+00  
 -9.08823529e+00 4.38207547e+00 5.27516779e+00 1.50000000e+00  
 8.15958904e+00 9.90821256e+00 -4.31587057e+00 3.71182266e+00  
 -2.03211009e+00 -5.50000000e+00 -1.60233918e+00 2.21546961e+00  
 3.96147920e+00 4.66485648e+00 -9.48203593e+00 1.72087379e+00  
 2.70783848e+00 -7.25000000e+00 4.14625850e+00 -3.25335570e+00  
 7.54729730e+00 6.85078534e+00 4.89570552e+00 7.26450276e+00  
 -6.22070312e-01 4.59789685e+00 7.23639191e+00 6.78691275e+00  
 2.54368132e+00 3.85467626e+00 -7.13935108e+00 2.55614973e-01  
 -1.03893715e+01 4.53181818e-01 -7.93312986e+00 2.00000000e+00  
 1.04072086e+01 1.04810776e+01 5.87733209e+00 -1.65076336e+00  
 -3.31030512e+00 3.16359164e+00 3.72933634e+00 7.49912536e+00  
 5.26194399e+00 9.96926230e+00 -6.78566094e+00 1.32259207e+00  
 -6.41077999e+00 1.26354909e-01 -1.13597406e+00 3.24352332e+00  
 -1.32753859e+01 3.92123288e+00 9.36321381e+00 6.77820451e+00  
 1.12385661e+00 -2.66444563e+00 -1.27520911e+01 6.25587393e+00  
 1.33714705e+01 5.95420099e+00 -6.19435637e+00 4.15135783e+00  
 2.56603774e+00 -5.43533123e+00 -1.07043682e+01 -3.02730658e+00  
 -8.87406015e+00 8.05555556e-01 9.54815016e+00 5.57378401e+00  
 5.22473361e+00 -6.49515973e+00 3.14597701e+00 -1.10000000e+01  
 7.05691057e-01 3.83574380e-01 -7.00000000e+00 -4.88785913e+00  
 2.21979212e+00 9.11349693e+00 -5.56330563e+00 -2.57741935e+00  
 1.76755952e+00 9.27850038e+00 -2.19612449e-01 -5.67251276e+00  
 5.39985344e+00 -2.52949724e+00 -8.12892348e+00 1.45387352e+01  
 -1.23718161e+00 1.15000000e+01 5.60416667e+00 -2.63789760e+00  
 -1.62428408e+00 1.40825688e+00 9.17337627e-01 -7.95745170e+00  
 4.47195538e+00 6.67982456e+00 6.24940968e+00 2.31005819e+00  
 -8.06941106e+00 4.17114094e+00 3.11644487e+00 4.84090909e+00  
 -3.28876273e+00 9.74093722e+00 3.40419770e+00 7.81613892e+00  
 -8.30541237e+00 9.30059524e-02 8.85628743e+00 -8.96086957e+00  
 -1.24655602e+01 -7.29589935e+00 -4.12237330e+00 2.83542601e+00  
 6.43725100e+00 0.00000000e+00 1.55624143e+00 7.27156690e+00  
 1.67886905e+00 9.14688301e+00 8.04122340e+00 -3.55724299e-01  
 -5.09871032e+00 4.36387782e+00 -1.19681373e+00 -1.17113462e+01  
 5.35022272e+00 -2.11343188e+00 -4.67273535e+00 1.80646960e+00  
 -4.10447761e-02 -9.41917752e+00 5.85902721e+00 3.96360153e+00  
 7.38455657e+00 2.29388422e+00 4.63563830e+00 3.02439024e+00  
 4.84691011e+00 -8.89035964e+00 6.62735405e+00 9.05157171e+00  
 -1.30000000e+01 1.50000000e+01 1.00000000e+01 -5.14112903e-01  
 -4.25000000e+00 7.70855905e+00 -1.56385449e+00 8.72202487e+00  
 -2.71497518e+00 2.76706827e+00 -3.79100000e+00 -6.00000000e+00  
 8.18331473e+00 1.21070076e+00 -2.27635542e+00 -6.01306620e+00  
 4.45592287e-01 6.52686308e+00 4.25905292e+00 1.22457143e+01  
 1.50728477e+01 -3.59476440e+00 -2.83206371e+00 7.61653772e+00  
 4.68907563e+00 7.93442623e+00 2.61443418e+01 1.13028351e+01  
 -2.06224385e+00 7.45738204e+00 8.18302829e+00 6.75840880e+00  
 -6.00710419e+00 -2.88121166e+00 4.50411969e+00 8.36342593e+00  
 -2.33919722e+00 4.60003818e-01 -2.88194444e+00 1.47482517e+01  
 3.79696970e+00 -2.49330940e+00 9.25462963e+00 3.85081744e+00  
 1.55593220e+00 4.27181097e+00 8.16641855e+00 3.93893130e+00  
 3.50148515e+00 4.74647887e+00 3.62092111e+00 -7.51734450e+00  
 2.72000000e+00 1.70794872e+01 1.02315864e+01 -4.00438048e+00  
 6.74385246e+00 1.05922285e+01 -2.65314441e+00 7.22492837e+00  
 7.86786962e+00 1.29855403e+01 -2.09642094e+00 3.40796020e-02  
 7.53538390e+00 5.00000000e+00 1.51566092e+01 1.72911964e+00  
 1.72732908e+00 7.33456005e+00 1.04603102e+01 4.23828125e+00  
 1.05194878e+01 5.66843827e+00 -4.12741228e+00 1.25397554e+01  
 1.28443691e+01 3.93786550e+00 9.06201550e+00 9.60563380e+00  
 -7.25286849e+00 9.93632075e+00 1.35000000e+01 5.80529226e+00  
 -1.07630326e+01 -2.22276941e+00 -5.80059172e+00 4.00171625e+00  
 8.72878229e+00 1.00430514e+01 1.06953291e+01 -2.27535497e+00  
 1.84533537e+01 4.16326210e+00 -7.61759371e+00 7.28289077e+00  
 -1.12959119e+01 6.75103734e+00 9.05789826e+00 -1.11042403e+00  
 6.33481894e+00 4.27659574e+00 4.12239583e+00 -6.51027640e+00  
 9.32288828e+00 3.92619926e+00 7.60259857e+00 5.13062409e+00  
 -7.34354968e+00 1.40694308e+01 4.75374532e+00 6.45534290e+00  
 -7.10526316e-01 2.00373832e+00 3.51535598e+00 4.39470588e+00  
 2.18785112e+00 7.75139925e+00 7.97872996e+00 6.95109078e-01  
 7.77734839e+00 -1.76617133e+00 6.82965451e+00 1.88104640e+00  
 -1.12706422e+01 6.86127168e+00 5.88523192e+00 3.38780488e+00  
 -3.41011236e+00 3.86184211e+00 6.56907895e+00 -6.80545836e-02  
 7.33454545e+00 -5.00097087e+00 -8.79562044e-01 -1.75164114e+00  
 -3.28898007e+00 -4.44911805e+00 -7.55081001e+00 5.82270169e+00  
 3.90573770e+00 -1.04489796e+01 5.78095238e+00 -3.10344828e-01  
 -1.23177269e+00 -1.82328907e+00 5.42330383e+00 6.76455479e+00  
 -9.82897033e+00 6.56189711e+00 5.15866667e+00 2.53583427e+01  
 3.05809129e+00 1.76509207e+01 1.41640625e+01 1.41508982e+01  
 4.99638728e+00 -1.10434462e+01 1.47913669e+01 9.79545455e+00  
 -3.25000000e+00 2.36570248e+00 5.16951788e+00 8.75000000e+00  
 -3.59194282e+00 -4.65347334e+00 9.72904412e+00 1.87708333e+01  
 5.50137994e+00 -8.81765068e+00 -5.40136054e+00 -9.02489627e+00  
 6.83061889e+00 -2.17429022e+00 -1.80045249e+00 9.63223140e+00  
 5.51577909e+00 2.72981100e+00 -7.14542079e+00 2.28060201e+01  
 3.60816327e+00 -7.69286704e+00 -1.82673267e+00 1.75693431e+00  
 2.67289073e-01 -1.00077586e+01 -2.72630231e+00 -9.64993804e+00  
 -5.23170732e+00 -7.41535714e+00 7.07211538e+00 5.46260870e+00  
 9.87500000e+00 3.60714286e+00 -4.49613900e+00 1.79411765e+00  
 3.99347015e+00 -7.70925110e-02 4.83756684e+00 1.75322997e+00  
 -6.71359223e+00 7.07975680e+00 3.58785047e+00 2.94578313e+00  
 -8.64595808e+00 6.72857143e+00 1.04094923e+01 4.36340206e+00  
 4.81726283e+00 9.35028791e+00 -3.48110721e+00 2.89486994e+00  
 1.53846154e+01 -9.85011507e+00 8.24193548e-01 1.12823062e-01  
 4.02874088e+00 4.25454545e+00 5.34385522e+00 7.29728318e+00  
 6.20356869e+00 5.27051282e+00 -8.66254195e+00 9.38521401e+00  
 -8.29752917e+00 6.07482394e+00 1.77533784e+00 -1.31532012e+01  
 3.97184567e+00 -1.18904983e+01 9.09884837e+00 -1.03474320e-01  
 5.57812500e+00 1.57797619e+01 4.62892670e+00 -1.07469267e+01  
 -1.03559783e+01 -2.07755682e+01 -1.63399281e+00 -8.82853899e+00  
 8.37969925e+00 -4.45682451e-01 -7.99109907e+00 1.02570533e+01  
 -2.23629490e+00 6.68592437e-01 1.39207048e+00 7.57964410e+00  
 3.22809278e+00 -1.86821400e+00 -5.36329201e+00 -7.38369305e+00  
 2.45860566e+00 1.49163569e+00 -9.02918070e-01 -4.79041916e-03  
 8.84742952e-01 6.72747748e+00 -1.75457227e+01 -1.60620221e+00  
 9.21754933e+00 6.27058111e+00 1.28734508e+00 2.55289256e+00  
 1.03033937e+01 4.24927114e+00 -1.58506261e+01 -9.64666667e+00  
 6.59708738e-01 4.11111111e+00 -9.13939052e+00 -7.86740506e+00  
 -2.81290323e+00 1.76146341e+00 -3.44875223e+00 4.21535326e+00  
 5.90989011e+00 -1.18177208e+01 1.13679245e-01 -5.12854610e+00  
 -8.83076923e-01 -1.12143975e+01 9.62962963e+00 -1.02885598e+01  
 -1.88881830e+00 4.43301105e+00 -1.78025641e+00 1.53445946e+01  
 5.19515931e+00 6.76781971e+00 -2.59322034e+00 1.22592807e+00  
 2.67352185e+00 3.31851464e+00 3.79634831e+00 -1.67395231e+01  
 -5.03913043e+00 3.14529221e+00 -7.79074074e+00 9.31781915e+00  
 -1.10934385e+01 1.24081238e+00 1.53759615e+01 3.42968750e+00  
 -1.04246324e+01 -2.12741935e+00 7.05512153e+00 -1.04206308e+01  
 -1.59536697e+01 -2.68292683e+00 -5.06138614e+00 4.41929134e+00  
 -5.40753012e+00 2.17129630e+00 -1.02289562e+01 -1.38561947e+01  
 3.96402550e+00 3.74324324e+00 -7.01164875e+00 -5.48932927e+00  
 -5.24250000e+00 7.38269231e+00 2.44358407e+00 1.31818182e+00  
 -3.47992081e+00]  
OAT[DegC]  
 2.500000 6  
 6.500000 6  
-3.000000 5  
 4.000000 5  
 1.000000 5  
 ..  
-5.242500 1  
 7.382692 1  
 2.443584 1  
 1.318182 1  
-2.682927 1  
Name: count, Length: 665, dtype: int64  
Unique values in column 'Generalized\_Weight':  
[4000. 3500. 3315.129812 3030.42168675 3765.17571885  
 3615.74074074 3775.33632287 3728.35426305 3875.58188743 3894.77167439  
 3843.00301694 3045.96298556 3032.63577479 3183.32470223 3777.14143426  
 3166.47058824 3806.00649351 3230.18292683 3756.42023346 3160.87357736  
 3349.71838198 3145.3125 3625.68807339 3613.38289963 3838.85017422  
 3614.60957179 3716.60649819 3738.23109843 3856.08203678 3833.81924198  
 3807.47126437 3869.02927581 3000. 3051.05156038 3446.42857143  
 3347.43875278 3570.14925373 3256.96594427 3266.57645467 3314.81481481  
 3336.41771315 3542.75092937 3211.40350877 3489.57388939 3492.49061327  
 3591.85132237]  
Generalized\_Weight  
4000.000000 455  
3500.000000 130  
3000.000000 85  
3315.129812 1  
3765.175719 1  
3030.421687 1  
3775.336323 1  
3728.354263 1  
3875.581887 1  
3894.771674 1  
3843.003017 1  
3045.962986 1  
3032.635775 1  
3615.740741 1  
3183.324702 1  
3777.141434 1  
3806.006494 1  
3166.470588 1  
3756.420233 1  
3160.873577 1  
3349.718382 1  
3230.182927 1  
3625.688073 1  
3613.382900 1  
3838.850174 1  
3614.609572 1  
3716.606498 1  
3738.231098 1  
3856.082037 1  
3145.312500 1  
3833.819242 1  
3807.471264 1  
3869.029276 1  
3051.051560 1  
3446.428571 1  
3347.438753 1  
3570.149254 1  
3256.965944 1  
3266.576455 1  
3314.814815 1  
3336.417713 1  
3542.750929 1  
3211.403509 1  
3489.573889 1  
3492.490613 1  
3591.851322 1  
Name: count, dtype: int64  
Unique values in column 'FCR':  
[7.87056421 7.6556213 7.63047534 6.84409014 6.79932199 6.32785543  
 6.1035898 6.0206168 5.89882309 5.74697331 5.6153505 5.47532713  
 5.4246063 5.25166094 4.97933798 4.85813278 4.83090338 4.81441826  
 4.80606667 4.67165224 4.65623221 4.56664081 4.5313431 4.39719569  
 4.3564648 4.28457038 4.1400172 4.13878417 4.03813995 4.01613873  
 3.96097513 3.91619606 3.8877776 3.87963209 3.80378984 3.77197183  
 3.62864942 3.57923174 3.57418961 3.47919547 3.38945692 3.20734339  
 3.20463728 3.20268233 3.19709283 3.07845056 3.01690495 2.9916994  
 2.98006914 2.94376005 2.91245791 2.85985822 2.76416976 2.71447633  
 2.65098328 2.61672894 2.61055453 2.59780111 2.56632175 2.38628646  
 2.2978639 2.17474027 2.08036323 2.0371484 1.80595693 1.77933579  
 1.47725838 1.31722236 1.30343341 1.27624099 1.27272653 1.24060748  
 1.23309824 1.17775601 1.17559015 1.14284709 1.03272764 1.023079  
 1.01726693 0.97379788 0.9586622 0.93682339 0.92497239 0.85763989  
 0.84072079 0.82747697 0.80042675 0.77283009 0.7663034 0.74770741  
 0.73999649 0.69084475 0.69021154 0.60753257 0.60060314 0.54384602  
 0.5235028 0.51831151 0.33439666 0.1528927 0.02761159 0.  
 nan]  
FCR  
0.000000 127  
7.870564 1  
7.630475 1  
6.844090 1  
6.799322 1  
 ...   
0.543846 1  
0.518312 1  
0.334397 1  
0.152893 1  
0.027612 1  
Name: count, Length: 102, dtype: int64  
Unique values in column 'HV Battery Power[Watts]':  
[ 7.68971257e+03 -3.93199048e+03 6.03038573e+03 6.13095831e+03  
 6.98327776e+03 1.67341663e+03 3.37328144e+03 1.36488613e+03  
 2.19681604e+03 -3.21506297e+03 -4.41046315e+03 -9.99918291e+03  
 1.79935163e+03 -7.56929052e+02 3.52508655e+03 -2.71493235e+02  
 1.05081517e+03 1.52441312e+03 -5.60534914e+03 -3.38013944e+03  
 1.59169890e+03 -1.78901404e+03 -1.80366724e+03 3.21504925e+03  
 3.54173416e+02 1.83047356e+03 -7.05205726e+03 1.95143106e+02  
 -2.01259557e+03 7.50800586e+02 3.43587039e+02 1.51567829e+03  
 6.63336157e+03 1.73014901e+03 9.84677449e+02 9.59238520e+02  
 -6.10004672e+01 2.22562540e+03 1.67401425e+03 3.17665982e+03  
 2.61910868e+03 6.59420962e+02 -6.80473481e+02 3.02847024e+03  
 4.02304547e+02 -1.31857207e+03 2.85725295e+03 2.02256379e+02  
 8.35866203e+02 -8.77610190e+03 1.17352486e+03 1.22713132e+03  
 -1.58041527e+03 -1.92867791e+03 -8.23342987e+03 -6.40802117e+02  
 1.21551140e+03 -8.62928427e+02 1.06267070e+03 5.94694795e+01  
 -1.99258846e+03 -4.83814749e+03 -7.45061037e+03 -2.80573430e+02  
 -8.66518661e+03 -9.02941760e+02 -3.82316288e+03 -7.10465611e+03  
 -8.93820450e+02 -2.40381826e+03 -1.11840228e+04 -6.51690741e+03  
 -6.38986055e+02 -4.40534585e+03 -5.25103811e+03 -4.67129796e+03  
 -5.69841682e+03 -9.05663085e+03 -8.10895543e+03 -6.31318316e+03  
 -6.01489147e+03 -4.88195261e+03 -5.94931727e+03 -8.08978296e+03  
 -7.87467462e+03 -7.93955946e+03 -2.99800180e+02 -1.84056166e+04  
 -1.36154145e+04 -3.05834698e+03 -2.90543763e+03 -7.89612154e+03  
 -4.84472937e+03 -5.60821166e+03 -4.54000663e+03 -8.19548169e+03  
 -1.65412393e+03 -1.41606682e+04 -1.05056469e+04 -7.02161236e+03  
 -1.07802456e+04 1.07274240e+03 6.22581401e+01 -3.46167770e+02  
 -1.05302376e+03 -1.19979867e+03 -1.21537130e+03 -1.75663319e+03  
 -1.78093902e+03 -2.13627826e+03 -2.58392447e+03 -2.64962944e+03  
 -2.79657802e+03 -3.17943814e+03 -3.31931781e+03 -3.48480630e+03  
 -3.62261714e+03 -3.66241207e+03 -3.75480590e+03 -4.12630693e+03  
 -4.35837159e+03 -4.38239317e+03 -4.49017361e+03 -4.55827494e+03  
 -5.04670820e+03 -5.10957130e+03 -5.18287343e+03 -5.36044871e+03  
 -5.44372705e+03 -5.54297248e+03 -5.63306950e+03 -5.64614110e+03  
 -5.81016477e+03 -5.82576706e+03 -5.83103227e+03 -5.98798886e+03  
 -6.00614361e+03 -6.04867093e+03 -6.37649618e+03 -6.48211115e+03  
 -6.74423619e+03 -6.80403792e+03 -6.86764782e+03 -7.00477156e+03  
 -7.03388556e+03 -7.19320273e+03 -7.27157180e+03 -7.32511537e+03  
 -7.38866551e+03 -7.46985017e+03 -7.49285212e+03 -7.49382535e+03  
 -7.52788696e+03 -7.55215697e+03 -7.61388431e+03 -7.63539400e+03  
 -7.66649592e+03 -7.66960095e+03 -7.68865660e+03 -7.69833128e+03  
 -7.73252038e+03 -7.77285250e+03 -7.85739554e+03 -7.95356605e+03  
 -7.97755449e+03 -8.01615049e+03 -8.03772066e+03 -8.05609819e+03  
 -8.05807311e+03 -8.05985091e+03 -8.06702750e+03 -8.09321349e+03  
 -8.11431346e+03 -8.11759240e+03 -8.14470962e+03 -8.22416533e+03  
 -8.29856037e+03 -8.56453111e+03 -8.57350988e+03 -8.76772823e+03  
 -8.96735805e+03 -9.07054433e+03 -9.32835509e+03 -9.68222629e+03  
 -9.68337638e+03 -9.79982694e+03 -9.85291752e+03 -1.01062709e+04  
 -1.02454114e+04 -1.02977265e+04 -1.06667301e+04 -1.07429920e+04  
 -1.07550888e+04 -1.08661460e+04 -1.08738528e+04 -1.11744567e+04  
 -1.12981853e+04 -1.13247675e+04 -1.16215582e+04 -1.16482310e+04  
 -1.16627411e+04 -1.17283849e+04 -1.18177809e+04 -1.20101801e+04  
 -1.21678229e+04 -1.23482048e+04 -1.24038015e+04 -1.24119778e+04  
 -1.29496787e+04 -1.32805282e+04 -1.37527908e+04 -1.38440933e+04  
 -1.42141555e+04 -1.43623456e+04 -1.46068536e+04 -1.47275738e+04  
 -1.47307796e+04 -1.54976155e+04 -1.58016739e+04 -1.60373757e+04  
 -1.67762183e+04 -1.68521901e+04 -1.69844411e+04 -1.71686890e+04  
 -1.74424402e+04 -1.76889803e+04 -1.84096213e+04 -1.90652815e+04  
 3.22484530e+03 2.90574966e+03 2.68156960e+03 1.96885254e+03  
 1.92551932e+03 1.90934997e+03 1.49816901e+03 1.40838572e+03  
 1.20652027e+03 1.18673845e+03 1.14757684e+03 1.04221664e+03  
 1.02306098e+03 1.01132438e+03 9.97139637e+02 9.48400626e+02  
 8.97118838e+02 8.17889529e+02 7.87866522e+02 7.82221308e+02  
 7.71005873e+02 7.52689558e+02 7.48812328e+02 7.01344344e+02  
 6.86657363e+02 6.57121806e+02 6.54140112e+02 6.35193234e+02  
 6.18417224e+02 5.88007384e+02 5.47028424e+02 5.09208284e+02  
 4.82477949e+02 4.69920499e+02 4.52164340e+02 4.49084972e+02  
 4.42119474e+02 4.23966011e+02 4.19933284e+02 4.16095179e+02  
 3.93267348e+02 3.89825059e+02 3.85142517e+02 3.59736996e+02  
 3.54348117e+02 3.45105593e+02 3.44122502e+02 3.42833956e+02  
 3.09894228e+02 3.09133914e+02 3.07517145e+02 2.98514545e+02  
 2.98066952e+02 2.95041236e+02 2.95002865e+02 2.87136410e+02  
 2.78663481e+02 2.76970784e+02 2.71468692e+02 2.69729993e+02  
 2.68459073e+02 2.64257182e+02 2.56570994e+02 2.39870852e+02  
 2.19416275e+02 2.12440838e+02 1.99124420e+02 1.93778106e+02  
 1.90299870e+02 1.85601973e+02 1.84946570e+02 1.82727008e+02  
 1.81678000e+02 1.80839371e+02 1.64284744e+02 1.54585166e+02  
 1.51884717e+02 1.37976109e+02 1.35158915e+02 1.28674434e+02  
 1.12935236e+02 1.06625420e+02 8.44251756e+01 8.23041759e+01  
 8.18947104e+01 8.06595282e+01 7.38359375e+01 5.01916451e+01  
 4.25540693e+01 4.22886608e+01 3.54455173e+01 1.19814871e+01  
 1.18816029e+00 -4.54222246e+00 -3.15885445e+01 -3.45662202e+01  
 -7.24810965e+01 -7.30605994e+01 -8.25584947e+01 -8.73000246e+01  
 -1.06497658e+02 -1.36254291e+02 -1.37597033e+02 -1.39260135e+02  
 -1.52175613e+02 -2.00082961e+02 -2.02870410e+02 -2.06516806e+02  
 -2.45258781e+02 -2.56369580e+02 -2.75942093e+02 -2.83267095e+02  
 -3.21368644e+02 -3.33711353e+02 -3.45267287e+02 -3.49936150e+02  
 -3.54656894e+02 -3.69899847e+02 -3.72118970e+02 -3.91059952e+02  
 -4.04813938e+02 -4.17619706e+02 -4.23916557e+02 -4.83703857e+02  
 -5.02212418e+02 -5.30938716e+02 -5.40367511e+02 -5.58961170e+02  
 -5.61513083e+02 -5.61700416e+02 -5.84327934e+02 -5.98211005e+02  
 -6.17409608e+02 -7.11078263e+02 -7.52331989e+02 -7.82824614e+02  
 -8.16326690e+02 -8.45613800e+02 -9.35845420e+02 -9.49500472e+02  
 -1.00019466e+03 -1.00274347e+03 -1.10651139e+03 -1.13476220e+03  
 -1.22952595e+03 -1.23177197e+03 -1.24147443e+03 -1.28027477e+03  
 -1.32133731e+03 -1.36194143e+03 -1.39757793e+03 -1.43286094e+03  
 -1.44588347e+03 -1.61249485e+03 -1.66768072e+03 -1.68875558e+03  
 -1.74589928e+03 -1.81613334e+03 -1.91590177e+03 -1.93643198e+03  
 -1.99310855e+03 -2.02612112e+03 -2.13185448e+03 -2.16205122e+03  
 -2.26104428e+03 -2.27123344e+03 -2.36683962e+03 -2.39281588e+03  
 -2.49304678e+03 -2.49551887e+03 -2.81980738e+03 -2.82599652e+03  
 -2.83637192e+03 -2.84486053e+03 -2.93099683e+03 -2.98582903e+03  
 -3.03070217e+03 -3.08619680e+03 -3.16991127e+03 -3.23356432e+03  
 -3.29393238e+03 -3.30258647e+03 -3.31944748e+03 -3.33539844e+03  
 -3.36970474e+03 -3.40546766e+03 -3.43015303e+03 -3.50415371e+03  
 -3.51356869e+03 -3.53215399e+03 -3.63947237e+03 -3.67264066e+03  
 -3.73554237e+03 -3.79234938e+03 -3.87598855e+03 -3.88901073e+03  
 -3.88993189e+03 -3.98700176e+03 -4.08071517e+03 -4.14530678e+03  
 -4.17850250e+03 -4.18148987e+03 -4.20733888e+03 -4.21524077e+03  
 -4.22219316e+03 -4.28480795e+03 -4.29118835e+03 -4.30459599e+03  
 -4.31863359e+03 -4.34646081e+03 -4.40470678e+03 -4.44830889e+03  
 -4.46813445e+03 -4.48597240e+03 -4.49598256e+03 -4.53551078e+03  
 -4.54358478e+03 -4.55986255e+03 -4.58585070e+03 -4.59486243e+03  
 -4.75965169e+03 -4.76904597e+03 -4.77955290e+03 -4.79741500e+03  
 -4.81023559e+03 -4.83684034e+03 -4.84699093e+03 -4.84705030e+03  
 -4.87176408e+03 -4.87210572e+03 -4.88802417e+03 -4.94703083e+03  
 -5.05644030e+03 -5.08985273e+03 -5.10549110e+03 -5.12210119e+03  
 -5.13324687e+03 -5.14028230e+03 -5.14682901e+03 -5.21676721e+03  
 -5.24825619e+03 -5.24945664e+03 -5.26419336e+03 -5.27163787e+03  
 -5.41264907e+03 -5.44024662e+03 -5.47347958e+03 -5.48552729e+03  
 -5.55574834e+03 -5.59617624e+03 -5.60050864e+03 -5.64276811e+03  
 -5.64836181e+03 -5.65503925e+03 -5.66550477e+03 -5.67391605e+03  
 -5.70600078e+03 -5.70942017e+03 -5.73902239e+03 -5.77423267e+03  
 -5.82858614e+03 -5.83158028e+03 -5.85230502e+03 -5.85866904e+03  
 -5.86471324e+03 -5.86724735e+03 -5.89159394e+03 -5.90364141e+03  
 -5.92647807e+03 -5.94753264e+03 -5.95797012e+03 -5.96911457e+03  
 -5.97335564e+03 -5.97684218e+03 -6.00578151e+03 -6.04558005e+03  
 -6.13109564e+03 -6.15673936e+03 -6.17668750e+03 -6.19340719e+03  
 -6.22822122e+03 -6.24919265e+03 -6.37799833e+03 -6.38704175e+03  
 -6.40075164e+03 -6.41278556e+03 -6.42357013e+03 -6.43883072e+03  
 -6.45952135e+03 -6.48958964e+03 -6.50301352e+03 -6.50503230e+03  
 -6.52227893e+03 -6.52627440e+03 -6.54125721e+03 -6.54882242e+03  
 -6.58658657e+03 -6.60074494e+03 -6.60283968e+03 -6.61768935e+03  
 -6.62681054e+03 -6.63316600e+03 -6.63850487e+03 -6.64788589e+03  
 -6.67113246e+03 -6.67301004e+03 -6.68768653e+03 -6.69104697e+03  
 -6.70231027e+03 -6.70542518e+03 -6.70788325e+03 -6.71583089e+03  
 -6.71834814e+03 -6.72142479e+03 -6.75516376e+03 -6.75696162e+03  
 -6.79118043e+03 -6.82096809e+03 -6.88323693e+03 -6.92926696e+03  
 -6.95415731e+03 -6.97863636e+03 -6.98249046e+03 -6.98653741e+03  
 -7.01470038e+03 -7.05551294e+03 -7.06207361e+03 -7.07122595e+03  
 -7.09302986e+03 -7.12052922e+03 -7.14322640e+03 -7.15588210e+03  
 -7.17101949e+03 -7.20201307e+03 -7.21050663e+03 -7.21339095e+03  
 -7.23157411e+03 -7.23337973e+03 -7.30366759e+03 -7.32958390e+03  
 -7.33132009e+03 -7.35414582e+03 -7.35591957e+03 -7.37109604e+03  
 -7.39287101e+03 -7.42936255e+03 -7.42944238e+03 -7.46848782e+03  
 -7.47320192e+03 -7.53932175e+03 -7.57222794e+03 -7.59733005e+03  
 -7.60844930e+03 -7.61405522e+03 -7.64001615e+03 -7.65954909e+03  
 -7.70031075e+03 -7.70207877e+03 -7.70280294e+03 -7.70519067e+03  
 -7.76024667e+03 -7.76927524e+03 -7.77400586e+03 -7.81434193e+03  
 -7.82201916e+03 -7.86209069e+03 -7.88179390e+03 -7.88206485e+03  
 -7.89654949e+03 -7.95744823e+03 -7.97662708e+03 -8.04953812e+03  
 -8.05323675e+03 -8.06458279e+03 -8.10351430e+03 -8.10884548e+03  
 -8.11523865e+03 -8.13948665e+03 -8.18988959e+03 -8.22437184e+03  
 -8.31252608e+03 -8.32150334e+03 -8.33160557e+03 -8.34735099e+03  
 -8.36358950e+03 -8.36889885e+03 -8.38284653e+03 -8.40743846e+03  
 -8.42452993e+03 -8.45256826e+03 -8.46669712e+03 -8.49975837e+03  
 -8.50475310e+03 -8.52333170e+03 -8.54545519e+03 -8.55750648e+03  
 -8.57618921e+03 -8.64724335e+03 -8.71428807e+03 -8.72305670e+03  
 -8.72800939e+03 -8.73325352e+03 -8.75680647e+03 -8.78562695e+03  
 -8.81630336e+03 -8.83469515e+03 -8.84793435e+03 -8.90654294e+03  
 -8.93260709e+03 -8.98146583e+03 -9.05075997e+03 -9.09453080e+03  
 -9.09746004e+03 -9.10169443e+03 -9.14462845e+03 -9.17722337e+03  
 -9.24136962e+03 -9.29794031e+03 -9.43847300e+03 -9.45611149e+03  
 -9.54595409e+03 -9.55645696e+03 -9.62321244e+03 -9.63740671e+03  
 -9.75161088e+03 -9.75987882e+03 -9.89691868e+03 -9.99530403e+03  
 -1.00017358e+04 -1.00747144e+04 -1.02186436e+04 -1.02560790e+04  
 -1.02735985e+04 -1.03140555e+04 -1.03412007e+04 -1.04049443e+04  
 -1.04863975e+04 -1.04867794e+04 -1.05042159e+04 -1.06091068e+04  
 -1.06345906e+04 -1.06938063e+04 -1.08274230e+04 -1.09160333e+04  
 -1.10281605e+04 -1.10430127e+04 -1.10923021e+04 -1.12301403e+04  
 -1.12622117e+04 -1.13661162e+04 -1.14398437e+04 -1.15604357e+04  
 -1.16097671e+04 -1.17116806e+04 -1.20663017e+04 -1.21190611e+04  
 -1.21420369e+04 -1.23471647e+04 -1.24266955e+04 -1.26324160e+04  
 -1.27084367e+04 -1.27329033e+04 -1.27543221e+04 -1.28405980e+04  
 -1.28598482e+04 -1.32383203e+04 -1.33161754e+04 -1.33264959e+04  
 -1.33465642e+04 -1.35265244e+04 -1.38575073e+04 -1.40510691e+04  
 -1.41385466e+04 -1.42960929e+04 -1.44945044e+04 -1.49180212e+04  
 -1.50369362e+04 -1.56747097e+04 -1.62369566e+04 -1.65219224e+04  
 -1.69151223e+04 -1.71552923e+04 -1.73248551e+04 -1.90106048e+04  
 -2.31728968e+04]  
HV Battery Power[Watts]  
-23172.896785 1  
 7689.712572 1  
-3931.990481 1  
 6030.385734 1  
 6130.958314 1  
 ..  
-756.929052 1  
 1799.351626 1  
-9999.182915 1  
-4410.463150 1  
-3215.062975 1  
Name: count, Length: 713, dtype: int64  
Unique values in column 'MAF[g/sec]':  
[ nan 0.70999998 7.09815671 0.93836793 7.58933377 5.74408499  
 7.86608368 11.35131026 10.70294766 8.48524165 0.42999998 5.45044968  
 4.42776451 5.56597698 10.03554333 6.66707817 9.90607405 6.3045939  
 8.69913483 6.89304758 5.55368492 0.17 3.29644372 0.17022059  
 2.5424766 0.70999998 7.04854557 0.87 3.63488428 0.70999998  
 0.87 1.86524014 0.42999998 5.41128591 19.61492825 7.1385033  
 10.23973131 10.24601335 9.88047104 10.89331265 6.42225815 6.55019514  
 6.5065647 4.47597192 7.63244292 6.03637575 8.25817562 5.54300355  
 7.68490632 8.80207967 6.35581268 5.70124156 7.26172712 8.03390659  
 6.3455284 10.29756106 5.91053225 5.19591405 4.81290067 9.71809504  
 6.43029349 12.81992594 5.38840917 5.78868209 5.1244315 5.03909819  
 7.50713201 6.81243257 5.9920823 5.97637026 7.00919578 4.01926096  
 7.98378404 5.72352319 5.96161623 11.00420772 5.73099989 6.45165988  
 4.93275074 5.65230148 10.27395498 9.52212802 5.72611448 6.13437097  
 6.31001304 6.34264094 6.20818803 5.24660531 7.37547154 9.92433205  
 4.85518971 5.19013021 11.44695454 8.66693468 7.88766031 3.31986596  
 5.61154976 8.39005791 5.88159373 7.41805209 7.07443884 4.96072716  
 6.28277015 6.6288939 6.90692239 7.08300115 6.18562452 4.88490831  
 16.21632224 14.15733299 5.53484304 5.41580729 5.43294629 4.6062725  
 5.69406707 5.34427191 5.4358987 6.71206707 5.83938129 7.06727027  
 9.88525233 5.63974386 4.23926681 7.7018653 5.5972329 5.58187319  
 8.75377167 5.8158474 4.3091885 9.35781556 7.42994165 8.39721139  
 6.24249427 6.1979181 3.26798688 7.49740001 5.81428213 6.50120057  
 7.6489565 6.94492875 7.0107455 10.77566986 5.43336294 8.0952094  
 9.34352153 10.99888772 8.6663745 6.30396773 6.97129132 6.44896407  
 7.28988123 5.77597476 7.70825766 7.97701857 8.46290461 7.18073425  
 6.33749988 5.2145793 8.52687979 5.32766255 8.16126454 6.25400755  
 4.96855223 9.16264762 11.18742429 8.32223072 2.97617906 7.10236953  
 5.56281108 6.37814162 5.66938822 8.30749748 8.25356365 6.88771323  
 8.52260721 9.62280576 9.64275024 4.2961604 8.47903206 6.37673668  
 8.52968561 6.37329705 5.65836576 8.42181448 12.2057199 40.22999954  
 6.41232127 7.27496197 7.68545166 5.16876876 10.35787857 2.55746962  
 2.85933143 0.17005236 10.15455657 6.0065772 8.34668168 5.10921075  
 3.55732064 3.2816625 4.9914073 6.61062424 4.58964479 9.34329614  
 2.87787832 3.78771581 5.16574896 9.06249489 2.31539854 5.49759157  
 4.46832256 7.87135304 5.87572605 3.78450576 2.78308389 0.28415789  
 9.55438816 5.52517529 8.31510132 3.42993193 3.63869053 2.97370111  
 2.07949649 2.16891468 0.1700738 1.60567937 4.82546186 15.23246035  
 4.69148416 0.87 0.43607008 2.76473113 3.13011768 7.11142035  
 1.91780546 6.11412831 1.26316264 1.56846712 3.15476352 0.17029491  
 3.64091832 2.7396211 2.38404602 0.42999998 2.88327681 2.11087089  
 2.71834301 2.34905282 7.13627924 0.77757871 2.33081712 6.6033283  
 3.18150753 4.50067229 1.4282237 2.56479547 7.64689984 3.11230428  
 5.26898724 0.81812903 3.78182892 6.8012863 2.76312428 2.94098226  
 4.90742437 3.6189363 7.80997664 3.24657383 4.35157091 2.37070678  
 2.63859095 5.1760869 7.95804276 6.95345304 7.22317925 3.62770892  
 3.96510963 1.3016286 5.72389969 6.87587006 2.69551201 3.99732223  
 2.03285921 2.04675724 3.22825309 2.06226376 4.89032599 0.8902267  
 2.70706598 2.25361534 0.72591547 1.4162206 1.53697432 5.96220331  
 1.27471095 2.5478679 3.50284566 3.42129561 1.22970586 3.646258  
 2.42744787 1.47174308 11.45094468 1.54765056 2.78665847 1.64678732]  
MAF[g/sec]  
0.710000 89  
0.870000 21  
0.870000 6  
0.870000 4  
0.710000 4  
 ..  
5.723900 1  
1.301629 1  
3.965110 1  
3.627709 1  
0.938368 1  
Name: count, Length: 305, dtype: int64  
Unique values in column 'Absolute Load[%]':  
[nan]  
Series([], Name: count, dtype: int64)  
Unique values in column 'Short Term Fuel Trim Bank 1[%]':  
[nan]  
Series([], 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]  
Series([], 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']  
  
'''

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

'''  
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] 1850  
Heater Power[Watts] 1850  
Vehicle Speed[km/h] 2  
Distance[km] 2  
Engine RPM[RPM] 2  
OAT[DegC] 1148  
Generalized\_Weight 92  
FCR 212  
HV Battery Power[Watts] 1850  
MAF[g/sec] 207  
Absolute Load[%] 156  
Short Term Fuel Trim Bank 1[%] 80  
Short Term Fuel Trim Bank 2[%] 833  
Long Term Fuel Trim Bank 1[%] 87  
Long Term Fuel Trim Bank 2[%] 852  
dtype: int64  
  
Missing values in HEV Vehicles:  
Air Conditioning Power[Watts] 1164  
Heater Power[Watts] 1164  
Vehicle Speed[km/h] 1  
Distance[km] 1  
Engine RPM[RPM] 1  
OAT[DegC] 631  
FCR 82  
HV Battery Power[Watts] 1164  
MAF[g/sec] 7  
Absolute Load[%] 6  
Short Term Fuel Trim Bank 1[%] 74  
Short Term Fuel Trim Bank 2[%] 1068  
Long Term Fuel Trim Bank 1[%] 78  
Long Term Fuel Trim Bank 2[%] 1079  
dtype: int64  
  
Missing values in EV Vehicles:  
Engine RPM[RPM] 95  
FCR 95  
MAF[g/sec] 95  
Absolute Load[%] 95  
Short Term Fuel Trim Bank 1[%] 95  
Short Term Fuel Trim Bank 2[%] 95  
Long Term Fuel Trim Bank 1[%] 95  
Long Term Fuel Trim Bank 2[%] 95  
dtype: int64  
  
Missing values in PHEV Vehicles:  
Air Conditioning Power[Watts] 184  
Heater Power[Watts] 618  
FCR 390  
MAF[g/sec] 184  
Absolute Load[%] 618  
Short Term Fuel Trim Bank 1[%] 618  
Short Term Fuel Trim Bank 2[%] 618  
Long Term Fuel Trim Bank 1[%] 618  
Long Term Fuel Trim Bank 2[%] 618  
dtype: int64

# 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(),  
 'Vehicle Speed[km/h]': df\_ICE['Vehicle Speed[km/h]'].mean(),  
 'Distance[km]': df\_ICE['Distance[km]'].mean(),  
 'Engine RPM[RPM]': df\_ICE['Engine RPM[RPM]'].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(),'Vehicle Speed[km/h]': df\_ICE['Vehicle Speed[km/h]'].mean(),  
 'Distance[km]': df\_HEV['Distance[km]'].mean(),  
 'Engine RPM[RPM]': df\_HEV['Engine RPM[RPM]'].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)

df\_ICE.describe()

Trip Latitude[deg] Longitude[deg] \  
count 1850.000000 1850.000000 1850.000000   
mean 1104.844865 42.270912 -83.730017   
std 630.409547 0.014450 0.024162   
min 5.000000 42.228167 -83.799060   
25% 597.250000 42.261734 -83.743438   
50% 1066.500000 42.271157 -83.729257   
75% 1560.750000 42.280541 -83.714530   
max 2898.000000 42.317667 -83.676301   
  
 Air Conditioning Power[Watts] Heater Power[Watts] \  
count 1850.0 0.0   
mean 0.0 NaN   
std 0.0 NaN   
min 0.0 NaN   
25% 0.0 NaN   
50% 0.0 NaN   
75% 0.0 NaN   
max 0.0 NaN   
  
 Vehicle Speed[km/h] Distance[km] Engine RPM[RPM] OAT[DegC] \  
count 1850.000000 1850.000000 1850.000000 1850.000000   
mean 39.015194 3.309058 1419.302311 9.775714   
std 11.758976 2.884161 211.698176 7.752689   
min 2.315187 0.138885 672.498164 -37.615797   
25% 32.237114 1.779731 1294.994079 3.455565   
50% 37.594433 2.759217 1394.228477 15.000000   
75% 44.196954 4.063893 1510.973719 15.000000   
max 104.139269 45.644997 2696.845238 23.299127   
  
 Generalized\_Weight FCR HV Battery Power[Watts] MAF[g/sec] \  
count 1850.000000 1850.000000 1850.0 1850.000000   
mean 3493.050820 0.873750 0.0 12.391077   
std 554.956111 0.228333 0.0 3.437400   
min 2500.000000 0.320519 0.0 4.235972   
25% 3097.581051 0.731963 0.0 10.277281   
50% 3500.000000 0.867654 0.0 12.151134   
75% 3746.313644 0.954025 0.0 13.496370   
max 6000.000000 2.298379 0.0 31.219119   
  
 Absolute Load[%] Short Term Fuel Trim Bank 1[%] \  
count 1850.000000 1850.000000   
mean 31.495418 0.417175   
std 4.730420 2.115002   
min 17.194966 -7.173713   
25% 28.595927 -0.581242   
50% 31.495418 0.099822   
75% 33.783841 0.926757   
max 65.068105 17.910560   
  
 Short Term Fuel Trim Bank 2[%] Long Term Fuel Trim Bank 1[%] \  
count 1850.000000 1850.000000   
mean -0.025516 1.531788   
std 3.133955 3.710839   
min -82.059717 -16.385135   
25% -0.398955 -0.506914   
50% -0.025516 1.510724   
75% -0.025516 3.590080   
max 27.945243 23.878614   
  
 Long Term Fuel Trim Bank 2[%]   
count 1850.000000   
mean 1.545437   
std 3.170008   
min -10.732110   
25% 0.789335   
50% 1.545437   
75% 1.621051   
max 24.087992

df\_HEV.describe()

Trip Latitude[deg] Longitude[deg] \  
count 1164.000000 1164.000000 1164.000000   
mean 983.212199 42.271584 -83.732861   
std 515.772796 0.017908 0.027968   
min 8.000000 42.228755 -83.799822   
25% 603.750000 42.260316 -83.751087   
50% 939.500000 42.273864 -83.730601   
75% 1289.500000 42.283531 -83.711190   
max 2932.000000 42.316757 -83.679435   
  
 Air Conditioning Power[Watts] Heater Power[Watts] \  
count 1164.0 1164.0   
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   
  
 Vehicle Speed[km/h] Distance[km] Engine RPM[RPM] OAT[DegC] \  
count 1164.000000 1164.000000 1164.000000 1164.000000   
mean 44.398181 3.449325 1090.477207 8.854155   
std 11.139853 3.189938 262.692593 7.583781   
min 0.000000 0.000000 265.103448 -13.000000   
25% 37.914713 1.616139 915.507061 2.092175   
50% 43.209330 2.870381 1071.343876 15.000000   
75% 49.451699 4.346975 1253.673305 15.000000   
max 96.011730 53.547970 2546.437500 17.314754   
  
 Generalized\_Weight FCR HV Battery Power[Watts] MAF[g/sec] \  
count 1164.000000 1164.000000 1164.0 1164.000000   
mean 3236.013017 0.566654 0.0 8.335685   
std 401.061570 0.182241 0.0 2.696189   
min 2000.000000 0.188280 0.0 2.775069   
25% 3000.000000 0.463180 0.0 6.784858   
50% 3090.897992 0.543431 0.0 7.835598   
75% 3500.000000 0.617360 0.0 9.159218   
max 5000.000000 2.062937 0.0 29.435711   
  
 Absolute Load[%] Short Term Fuel Trim Bank 1[%] \  
count 1164.000000 1164.000000   
mean 27.302309 -0.671801   
std 8.164484 1.263061   
min 8.867478 -7.945980   
25% 23.384086 -1.157698   
50% 26.880582 -0.557701   
75% 30.394540 -0.075793   
max 154.924407 7.597805   
  
 Short Term Fuel Trim Bank 2[%] Long Term Fuel Trim Bank 1[%] \  
count 1164.000000 1164.000000   
mean 0.032357 -0.127626   
std 0.387498 2.166047   
min -3.556068 -7.110596   
25% 0.032357 -1.487380   
50% 0.032357 -0.127626   
75% 0.032357 1.175747   
max 4.540215 9.791153   
  
 Long Term Fuel Trim Bank 2[%]   
count 1164.000000   
mean -1.331489   
std 0.532416   
min -6.281250   
25% -1.331489   
50% -1.331489   
75% -1.331489   
max 3.458137

df\_PHEV.describe()

Trip Latitude[deg] Longitude[deg] \  
count 618.000000 618.000000 618.000000   
mean 1027.393204 42.273889 -83.723610   
std 579.918702 0.019057 0.025063   
min 2.000000 42.229778 -83.799610   
25% 590.000000 42.259793 -83.739610   
50% 955.500000 42.277656 -83.719121   
75% 1457.750000 42.286126 -83.707332   
max 2432.000000 42.319032 -83.678096   
  
 Air Conditioning Power[Watts] Heater Power[Watts] \  
count 618.000000 618.0   
mean 27.315283 0.0   
std 165.769118 0.0   
min 0.000000 0.0   
25% 0.000000 0.0   
50% 0.000000 0.0   
75% 0.000000 0.0   
max 2008.721360 0.0   
  
 Vehicle Speed[km/h] Distance[km] Engine RPM[RPM] OAT[DegC] \  
count 618.000000 618.000000 618.000000 618.000000   
mean 42.794632 3.630850 452.272555 1.400766   
std 12.024760 3.956138 482.111349 6.943586   
min 14.977123 0.045077 0.000000 -20.775568   
25% 35.673644 1.383368 0.000000 -3.000000   
50% 41.934884 2.855270 346.214084 2.500000   
75% 47.761296 4.189252 783.824850 6.237949   
max 101.025354 51.932325 3360.000000 26.144342   
  
 Generalized\_Weight FCR HV Battery Power[Watts] MAF[g/sec] \  
count 618.000000 618.000000 618.000000 618.000000   
mean 3800.047327 1.310643 -5064.759049 4.287581   
std 366.069026 1.188460 5028.038070 3.193834   
min 3000.000000 0.000000 -23172.896785 0.170000   
25% 3806.372686 1.191592 -8178.594596 1.419221   
50% 4000.000000 1.310643 -5249.647151 4.287581   
75% 4000.000000 1.310643 -287.400366 5.717953   
max 4000.000000 7.870564 7689.712572 40.230000   
  
 Absolute Load[%] Short Term Fuel Trim Bank 1[%] \  
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   
  
 Short Term Fuel Trim Bank 2[%] Long Term Fuel Trim Bank 1[%] \  
count 618.0 0.0   
mean 0.0 NaN   
std 0.0 NaN   
min 0.0 NaN   
25% 0.0 NaN   
50% 0.0 NaN   
75% 0.0 NaN   
max 0.0 NaN   
  
 Long Term Fuel Trim Bank 2[%]   
count 618.0   
mean 0.0   
std 0.0   
min 0.0   
25% 0.0   
50% 0.0   
75% 0.0   
max 0.0

df\_EV.describe()

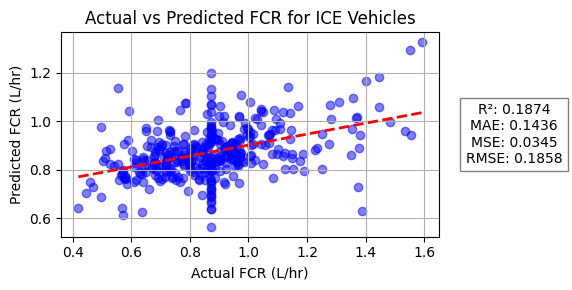
Trip Latitude[deg] Longitude[deg] \  
count 95.000000 95.000000 95.000000   
mean 1274.105263 42.271297 -83.735505   
std 533.419428 0.017869 0.023455   
min 554.000000 42.231394 -83.792418   
25% 737.000000 42.260824 -83.749556   
50% 1149.000000 42.272546 -83.736166   
75% 1806.000000 42.284664 -83.723986   
max 2165.000000 42.307465 -83.682248   
  
 Air Conditioning Power[Watts] Heater Power[Watts] \  
count 95.000000 95.000000   
mean 451.206427 595.948098   
std 389.532228 766.175778   
min 0.000000 0.000000   
25% 79.973193 0.000000   
50% 370.610687 307.565789   
75% 785.391036 852.601621   
max 1224.153166 3545.282369   
  
 Vehicle Speed[km/h] Distance[km] Engine RPM[RPM] OAT[DegC] \  
count 95.000000 95.000000 95.0 95.000000   
mean 38.120463 2.529701 0.0 2.758068   
std 12.568234 2.356622 0.0 6.301469   
min 3.490542 0.174693 0.0 -13.153201   
25% 30.045382 0.879309 0.0 -2.118267   
50% 39.347546 1.704800 0.0 2.894870   
75% 45.667038 3.458791 0.0 7.279737   
max 67.570289 11.031075 0.0 15.156609   
  
 Generalized\_Weight FCR HV Battery Power[Watts] MAF[g/sec] \  
count 95.0 95.0 95.000000 95.0   
mean 3500.0 0.0 -6257.533768 0.0   
std 0.0 0.0 2555.379815 0.0   
min 3500.0 0.0 -13857.507261 0.0   
25% 3500.0 0.0 -7292.243492 0.0   
50% 3500.0 0.0 -6377.998333 0.0   
75% 3500.0 0.0 -4515.746667 0.0   
max 3500.0 0.0 73.835938 0.0   
  
 Absolute Load[%] Short Term Fuel Trim Bank 1[%] \  
count 95.0 95.0   
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 95.0 95.0   
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 95.0   
mean 0.0   
std 0.0   
min 0.0   
25% 0.0   
50% 0.0   
75% 0.0   
max 0.0

"""  
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]: 0.0095  
 Distance[km]: 0.0003  
  
Regression Equation:  
 Slope of the regression line: [0.00948307 0.0002586 ]  
 Intercept: 0.5035  
 Target Variable: FCR  
 FCR = 0.0095\*Vehicle Speed[km/h] + 0.0003\*Distance[km] + 0.5035  
  
Evaluation Metrics:  
 R² Score : 0.1874  
 MAE : 0.1436  
 MSE : 0.0345  
 RMSE : 0.1858



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

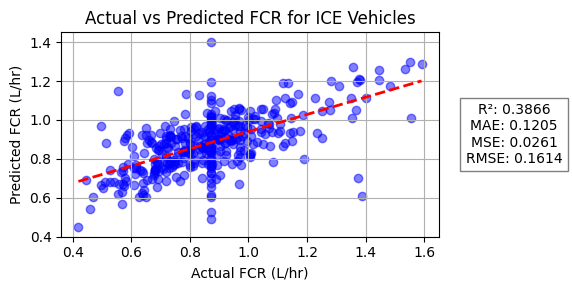
Vehicle Speed[km/h] Distance[km]  
3086 71.199588 3.336882  
339 55.280435 1.337339  
422 55.131579 3.372505  
2801 78.166852 5.286740  
3480 36.728477 0.398852  
3086 2.298379  
339 2.188477  
422 2.076016  
2801 1.994577  
3480 1.984249  
Name: FCR, dtype: float64

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]  
583 51.631868 3.327974  
113 33.507289 2.118764  
2559 36.015909 3.028503  
1002 43.093996 5.676134  
2331 30.373119 2.284008  
... ... ...  
3092 33.721195 2.308274  
647 37.289376 5.401196  
2829 27.263300 1.494319  
2343 42.880246 1.428060  
3164 31.380020 2.640701  
  
[370 rows x 2 columns]  
  
Parameter 2: 583 1.106431  
113 0.553769  
2559 0.942000  
1002 0.842179  
2331 0.717631  
 ...   
3092 0.965098  
647 0.812843  
2829 0.829558  
2343 1.069924  
3164 1.151980  
Name: FCR, Length: 370, dtype: float64  
  
Parameter 3: [0.9939539 0.82176452 0.8457892 0.91359587 0.7920857 0.64182502  
 0.98392366 0.83650246 0.85241939 0.85613979 0.75647866 0.83690364  
 0.88411493 0.81351054 0.82282677 1.01716732 0.81527195 0.9042472  
 0.88331503 0.82556969 0.89267805 0.78168545 0.86316732 1.19796438  
 0.85521821 0.95822974 0.76551403 0.82812902 0.84953921 0.87448858  
 0.84114759 0.8782002 0.74750913 0.61111284 0.8672889 0.81367471  
 0.86000763 0.66248591 0.94559838 1.06637243 0.89270525 1.06797823  
 0.92067015 0.7951683 1.04198101 0.89913217 1.10208613 1.03342681  
 0.82913852 0.78856374 0.90130866 0.80839429 0.91092538 0.77291493  
 0.83332047 0.74873194 0.91402483 0.77651454 0.96121827 0.94472434  
 0.7408862 0.77399781 1.00916142 0.69524308 0.88483323 1.06457409  
 0.9014151 0.94449262 0.85468957 0.88444692 0.80691125 0.90406515  
 1.06470471 0.83310985 1.07832545 1.01076311 1.02149514 0.85229187  
 0.83142444 0.90918657 0.8329109 0.95034766 0.80704741 0.85497727  
 0.71819912 0.87388064 0.96706136 0.88001538 0.96935849 0.83026783  
 0.83468426 0.78184435 0.77965881 0.81181064 0.76645376 0.81436996  
 1.14278585 0.86171505 0.89034212 0.84046658 0.9260068 0.74457636  
 0.88847415 0.93802494 0.84618808 0.81200129 0.90187641 0.94899367  
 0.85477967 0.87617821 0.79555537 0.80571784 0.99494568 0.85677095  
 0.89961695 0.72607188 0.79599739 1.1355591 0.92572353 0.80605597  
 1.00313919 1.04456797 0.90727261 0.83789001 0.84360605 1.04883482  
 1.01668756 1.03834764 0.94155141 0.8538893 1.07040484 0.88446227  
 0.99500933 0.85113445 0.76094029 0.87066293 0.87006935 0.83429824  
 0.83367925 0.89609742 1.32854091 0.84259497 0.62792602 0.79795852  
 0.80855921 0.68808692 0.87896082 0.85233315 0.96781173 0.83585466  
 0.80925996 0.86729781 0.97619347 0.7531233 0.82995685 0.82906413  
 0.871217 0.80283074 0.8235277 0.76438313 0.88631705 0.92608995  
 0.90723092 0.89233052 0.850361 0.87468441 0.71362916 1.07545202  
 0.74816529 0.79179872 0.79017028 0.87463591 0.83872342 0.89056412  
 0.90700891 0.96361647 0.8336606 0.79722955 0.79191604 0.96157745  
 0.75550984 0.63885065 0.77076683 0.94641812 0.89502058 0.86478517  
 0.90029063 0.74062409 1.10791807 1.18191916 0.8264435 0.81457423  
 0.96137056 0.95570408 0.98415995 0.98439991 0.95758834 0.90459268  
 0.92064088 0.85444354 0.81204673 0.85791739 0.90457045 0.71298415  
 0.82405351 0.72675322 0.86212867 1.16737675 0.92974894 0.68640163  
 1.09558374 0.86012993 0.86326442 0.8042157 0.72831643 0.79246755  
 0.73932466 0.90436025 0.76477432 0.79570542 0.82273752 0.84649848  
 0.897483 0.83605865 0.97066522 0.64361323 0.83121056 0.56151176  
 0.83923079 0.97018801 0.96246635 1.03927756 0.9674037 0.94464622  
 0.79049829 1.07816071 0.86267444 0.80885319 0.83456547 0.83308281  
 1.01475946 0.87428442 0.93808398 0.84654209 0.82089236 0.91550127  
 1.29430153 0.9072162 0.87363202 0.85189595 0.82583275 0.76586172  
 0.80258369 0.92769872 1.04527675 0.83779377 0.78952779 0.85349633  
 0.84622413 0.99641225 0.83680583 0.78072417 0.77878894 1.13446318  
 0.82756009 0.83441245 1.04039123 0.77100184 1.0218817 0.9424901  
 0.9437238 1.07528281 0.84672872 0.92070454 0.89303507 0.95347215  
 0.86880941 0.84684869 1.03043482 0.76188533 0.89217346 0.7513742  
 0.96897797 0.92256645 0.80169259 0.76614367 0.80678818 0.89783985  
 0.7781229 0.92869834 0.91745569 0.69993541 0.92630056 0.86951478  
 0.84133846 0.87780121 0.78755995 0.94698342 0.81588442 0.88738188  
 0.72176345 0.79405963 0.96291658 0.82796738 0.79448256 0.80245911  
 0.82977011 0.88040021 0.79849278 0.92385425 0.76688491 0.62648233  
 0.8519022 0.757303 0.90763319 0.93383115 0.84736665 0.7978643  
 0.81666045 0.91669207 0.69115113 0.96230292 0.75640201 1.00951639  
 0.63598123 1.02131319 0.84561509 0.86081006 0.84197648 0.82454573  
 0.75704037 0.87708701 1.01307439 0.92812791 0.86466055 1.0593637  
 0.86192691 0.95343892 0.9005557 0.94874232 0.76427784 0.88093342  
 0.9916847 1.03617148 0.82474121 0.87902633 0.90706791 0.77017091  
 0.84227401 0.9767083 0.75393782 0.81390641 0.84144057 0.96354838  
 0.89307189 0.70566034 0.81845904 0.90904024 0.7486833 0.88136834  
 0.91393472 0.67014678 0.97106744 0.86246727 0.74702507 0.82384202  
 0.85847916 0.76239085 0.91047033 0.80172645]  
  
Parameter 4: LinearRegression()  
  
Parameter 5: [0.41945073 0.44632772 0.45769057 0.46992443 0.49530756 0.49707377  
 0.50228967 0.50978674 0.51478663 0.52579723 0.52637484 0.54194688  
 0.553267 0.55376885 0.55764384 0.56482182 0.56806279 0.56971094  
 0.5711509 0.5717387 0.57324334 0.5738909 0.57422487 0.57830155  
 0.57880852 0.58186231 0.58507881 0.590104 0.59327404 0.60141015  
 0.60776458 0.60785096 0.60921732 0.61397063 0.61639979 0.61698189  
 0.62466832 0.62927278 0.6351436 0.63796808 0.63807282 0.63900412  
 0.63948399 0.64059093 0.64164955 0.64390014 0.64870657 0.64879215  
 0.64961685 0.65203972 0.6547778 0.65978573 0.66756689 0.66832681  
 0.66983155 0.67221448 0.67633889 0.6782872 0.6788129 0.68001115  
 0.68066674 0.68092637 0.68210614 0.6824217 0.68580104 0.68785613  
 0.68870445 0.68875587 0.68951611 0.69161696 0.6931242 0.69372443  
 0.69429729 0.69512881 0.69656279 0.70199021 0.70617919 0.70623401  
 0.70973559 0.71254186 0.71551087 0.71688475 0.71690339 0.71696905  
 0.71763093 0.71862693 0.7187749 0.71945953 0.7227016 0.72318853  
 0.72692299 0.72855976 0.7292131 0.73029928 0.73057853 0.73073431  
 0.73088625 0.73261176 0.7336237 0.73412722 0.73616271 0.73796329  
 0.74168284 0.74485876 0.74543707 0.74983061 0.74994332 0.75041332  
 0.75119877 0.75309006 0.75656973 0.75765363 0.7591541 0.7591757  
 0.76596516 0.76652022 0.76872322 0.77173658 0.77370336 0.7764197  
 0.77810851 0.77914931 0.77956346 0.77966226 0.78082811 0.78169299  
 0.78182816 0.78321825 0.78372075 0.78555108 0.79195471 0.79753415  
 0.79860257 0.79869195 0.7987673 0.79877254 0.7987929 0.79895328  
 0.8022148 0.80303254 0.80356523 0.80924968 0.81038562 0.81199781  
 0.81239504 0.81284277 0.8138821 0.81567464 0.82163163 0.82242376  
 0.82251032 0.82875279 0.82948298 0.82955765 0.83131277 0.83291398  
 0.83864942 0.83996348 0.84017546 0.84020314 0.8403431 0.84217886  
 0.8449058 0.84524656 0.84818987 0.85046469 0.85172323 0.85390144  
 0.85597616 0.85600133 0.85804911 0.85865434 0.85919767 0.85955769  
 0.86163629 0.86293011 0.86357834 0.8640867 0.86514297 0.86532019  
 0.86941103 0.86991574 0.86992652 0.87320828 0.87375027 0.87375027  
 0.87375027 0.87375027 0.87375027 0.87375027 0.87375027 0.87375027  
 0.87375027 0.87375027 0.87375027 0.87375027 0.87375027 0.87375027  
 0.87375027 0.87375027 0.87375027 0.87375027 0.87375027 0.87375027  
 0.87375027 0.87375027 0.87375027 0.87375027 0.87375027 0.87375027  
 0.87375027 0.87375027 0.87375027 0.87375027 0.87375027 0.87375027  
 0.87375027 0.87375027 0.87375027 0.87375027 0.87375027 0.87375027  
 0.87375027 0.87662668 0.87666632 0.88380318 0.88408621 0.88410634  
 0.88788022 0.89093141 0.89185446 0.89503246 0.89695331 0.89747579  
 0.89823355 0.90035961 0.90138535 0.90253082 0.90270855 0.90354091  
 0.90410709 0.90428348 0.9045019 0.90987403 0.91179687 0.91557109  
 0.91652845 0.91708579 0.91890475 0.91895271 0.92045819 0.92177784  
 0.92204562 0.92273382 0.9234484 0.92519212 0.9305476 0.9346588  
 0.93839178 0.93933963 0.93989779 0.94199985 0.94413705 0.94470638  
 0.94480483 0.94594288 0.94669477 0.94672675 0.95056866 0.9525868  
 0.9537955 0.95409933 0.95837657 0.96079219 0.96326402 0.96327626  
 0.96509829 0.96725296 0.96748784 0.96810866 0.96852742 0.96944215  
 0.97071009 0.97100208 0.97167369 0.97168144 0.97219156 0.97335311  
 0.97611708 0.9778503 0.98487865 0.98552181 0.98919694 0.99504151  
 0.99668228 0.99833343 1.00041272 1.00369501 1.00496536 1.00521975  
 1.00524143 1.00815644 1.0086286 1.01018741 1.0151582 1.01596901  
 1.02275503 1.02378816 1.02883876 1.04123676 1.04205105 1.04242758  
 1.04397119 1.04888419 1.04921254 1.05540021 1.06649367 1.06992375  
 1.0700806 1.07203007 1.08086024 1.08175213 1.08446847 1.08623787  
 1.09150023 1.09562953 1.09679748 1.10224179 1.10643068 1.10661808  
 1.11677049 1.11763674 1.12323102 1.12414032 1.13576927 1.14146213  
 1.14276383 1.14485966 1.14916688 1.15030852 1.15197964 1.1674869  
 1.17507275 1.18626755 1.18838147 1.22735341 1.23332794 1.24987914  
 1.24993139 1.27358503 1.27729844 1.28073581 1.30759027 1.35137645  
 1.3547024 1.35677714 1.36742265 1.37107883 1.37329207 1.37620079  
 1.38064729 1.38818105 1.39979044 1.44439209 1.44622684 1.48194045  
 1.5346745 1.55205487 1.55442818 1.59088871]  
  
Parameter 6: [0.77050138 0.77657199 0.77913848 0.7819017 0.7876349 0.78803383  
 0.78921192 0.79090526 0.79203457 0.7945215 0.79465196 0.79816916  
 0.800726 0.80083935 0.80171458 0.80333585 0.80406787 0.80444014  
 0.80476538 0.80489814 0.80523799 0.80538425 0.80545968 0.80638047  
 0.80649498 0.80718472 0.80791122 0.80904625 0.80976225 0.81159993  
 0.81303518 0.81305469 0.81336331 0.81443692 0.81498559 0.81511706  
 0.81685317 0.81789317 0.81921919 0.81985714 0.8198808 0.82009115  
 0.82019954 0.82044956 0.82068866 0.821197 0.82228261 0.82230194  
 0.82248821 0.82303546 0.8236539 0.82478502 0.82654253 0.82671417  
 0.82705404 0.82759226 0.82852383 0.82896389 0.82908262 0.82935327  
 0.82950134 0.82955999 0.82982646 0.82989773 0.83066101 0.83112519  
 0.8313168 0.83132841 0.83150012 0.83197463 0.83231507 0.83245064  
 0.83258003 0.83276784 0.83309173 0.8343176 0.83526375 0.83527614  
 0.83606703 0.83670087 0.83737147 0.83768178 0.83768599 0.83770082  
 0.83785032 0.83807528 0.8381087 0.83826334 0.83899561 0.8391056  
 0.83994909 0.84031878 0.84046634 0.84071168 0.84077475 0.84080993  
 0.84084425 0.84123399 0.84146255 0.84157628 0.84203603 0.84244272  
 0.84328284 0.84400018 0.8441308 0.84512315 0.84514861 0.84525477  
 0.84543217 0.84585935 0.84664529 0.84689011 0.84722902 0.84723389  
 0.84876741 0.84889278 0.84939036 0.85007098 0.85051521 0.85112874  
 0.85151018 0.85174527 0.85183881 0.85186112 0.85212445 0.8523198  
 0.85235033 0.8526643 0.8527778 0.85319121 0.85463758 0.85589779  
 0.85613911 0.85615929 0.85617632 0.8561775 0.8561821 0.85621832  
 0.85695499 0.85713969 0.85726001 0.85854393 0.85880051 0.85916464  
 0.85925437 0.85935549 0.85959024 0.85999512 0.8613406 0.86151952  
 0.86153907 0.86294904 0.86311396 0.86313083 0.86352725 0.86388891  
 0.86518435 0.86548116 0.86552904 0.86553529 0.8655669 0.86598154  
 0.86659746 0.86667443 0.86733922 0.86785303 0.86813729 0.86862928  
 0.86909788 0.86910357 0.86956609 0.8697028 0.86982552 0.86990683  
 0.87037632 0.87066855 0.87081496 0.87092978 0.87116836 0.87120839  
 0.87213237 0.87224637 0.8722488 0.87299005 0.87311246 0.87311246  
 0.87311246 0.87311246 0.87311246 0.87311246 0.87311246 0.87311246  
 0.87311246 0.87311246 0.87311246 0.87311246 0.87311246 0.87311246  
 0.87311246 0.87311246 0.87311246 0.87311246 0.87311246 0.87311246  
 0.87311246 0.87311246 0.87311246 0.87311246 0.87311246 0.87311246  
 0.87311246 0.87311246 0.87311246 0.87311246 0.87311246 0.87311246  
 0.87311246 0.87311246 0.87311246 0.87311246 0.87311246 0.87311246  
 0.87311246 0.87376215 0.8737711 0.87538308 0.87544701 0.87545155  
 0.87630395 0.87699311 0.87720159 0.8779194 0.87835325 0.87847126  
 0.87864242 0.87912262 0.8793543 0.87961303 0.87965317 0.87984117  
 0.87996905 0.8800089 0.88005823 0.88127161 0.88170592 0.88255839  
 0.88277462 0.88290051 0.88331135 0.88332218 0.88366222 0.88396029  
 0.88402077 0.88417621 0.88433761 0.88473146 0.88594108 0.88686966  
 0.88771282 0.88792691 0.88805298 0.88852776 0.88901048 0.88913908  
 0.88916131 0.88941836 0.88958819 0.88959541 0.89046317 0.890919  
 0.89119201 0.89126063 0.89222672 0.89277233 0.89333063 0.89333339  
 0.89374493 0.8942316 0.89428465 0.89442487 0.89451945 0.89472606  
 0.89501245 0.8950784 0.89523009 0.89523184 0.89534706 0.89560942  
 0.89623371 0.89662518 0.89821265 0.89835792 0.89918801 0.9005081  
 0.9008787 0.90125164 0.90172128 0.90246264 0.90274957 0.90280702  
 0.90281192 0.90347033 0.90357697 0.90392905 0.90505179 0.90523492  
 0.90676766 0.90700101 0.90814177 0.91094206 0.91112599 0.91121103  
 0.91155968 0.91266936 0.91274353 0.91414111 0.91664676 0.9174215  
 0.91745692 0.91789725 0.91989168 0.92009313 0.92070666 0.92110631  
 0.9222949 0.92322757 0.92349138 0.92472106 0.92566719 0.92570952  
 0.92800261 0.92819827 0.92946183 0.92966721 0.9322938 0.93357963  
 0.93387364 0.93434702 0.93531987 0.93557773 0.93595518 0.93945775  
 0.94117114 0.94369967 0.94417714 0.9529796 0.95432904 0.95806741  
 0.95807921 0.96342178 0.96426051 0.9650369 0.97110242 0.98099226  
 0.98174348 0.98221209 0.98461656 0.98544237 0.98594227 0.98659925  
 0.98760356 0.98930519 0.99192736 1.00200139 1.00241579 1.0104823  
 1.02239316 1.02631881 1.02685486 1.03509008]

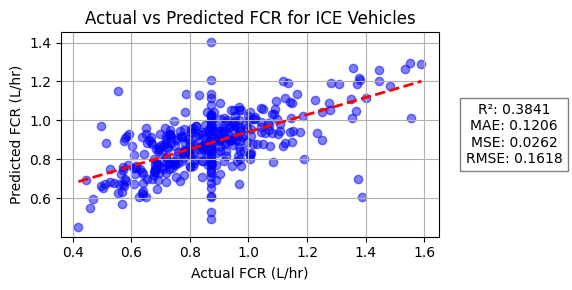
"""  
Scenario 2 for ICE  
Performed linear regression analysis using selected features to predict FCR for ICE vehicles.  
Plotted Actual vs Predicted FCR values and regression line.  
Displayed evaluation metrics: R², MAE, MSE, and RMSE on the plot.  
"""  
  
features = ['Latitude[deg]','Longitude[deg]','Vehicle 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]: 0.0605  
 Longitude[deg]: 0.1722  
 Vehicle Speed[km/h]: 0.0074  
 Distance[km]: -0.0015  
 Engine RPM[RPM]: 0.0002  
 OAT[DegC]: 0.0011  
 Generalized\_Weight: 0.0002  
  
Regression Equation:  
 Slope of the regression line: [ 0.06045161 0.17222124 0.00735495 -0.00152155 0.00022506 0.00105238  
 0.00020136]  
 Intercept: 11.4261  
 Target Variable: FCR  
 FCR = 0.0605\*Latitude[deg] + 0.1722\*Longitude[deg] + 0.0074\*Vehicle Speed[km/h] + -0.0015\*Distance[km] + 0.0002\*Engine RPM[RPM] + 0.0011\*OAT[DegC] + 0.0002\*Generalized\_Weight + 11.4261  
  
Evaluation Metrics:  
 R² Score : 0.3866  
 MAE : 0.1205  
 MSE : 0.0261  
 RMSE : 0.1614



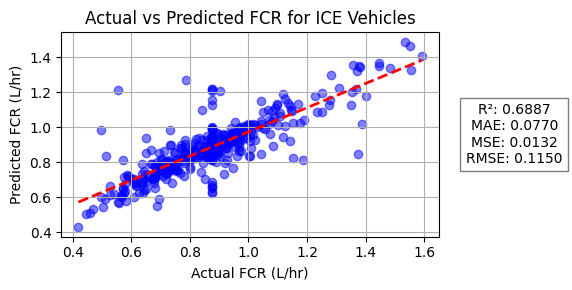
"""  
Scenario 3 for ICE  
Performed linear regression analysis using selected features to predict FCR for ICE vehicles.  
Plotted Actual vs Predicted FCR values and regression line.  
Displayed evaluation metrics: R², MAE, MSE, and RMSE on the plot.  
"""  
  
features = ['Vehicle Speed[km/h]', 'Distance[km]', '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]: 0.0074  
 Distance[km]: -0.0014  
 Engine RPM[RPM]: 0.0002  
 OAT[DegC]: 0.0010  
 Generalized\_Weight: 0.0002  
  
Regression Equation:  
 Slope of the regression line: [ 0.00735636 -0.00137758 0.0002257 0.00104617 0.0002006 ]  
 Intercept: -0.4373  
 Target Variable: FCR  
 FCR = 0.0074\*Vehicle Speed[km/h] + -0.0014\*Distance[km] + 0.0002\*Engine RPM[RPM] + 0.0010\*OAT[DegC] + 0.0002\*Generalized\_Weight + -0.4373  
  
Evaluation Metrics:  
 R² Score : 0.3841  
 MAE : 0.1206  
 MSE : 0.0262  
 RMSE : 0.1618



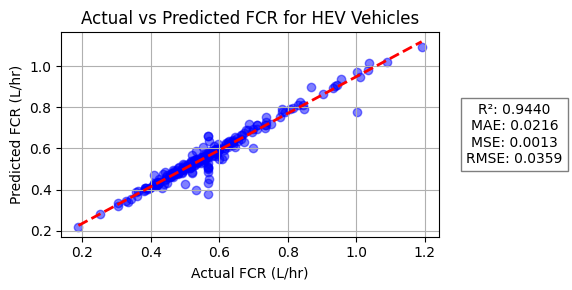
"""  
Scenario 4 for ICE  
Performed linear regression analysis using selected features to predict FCR for ICE vehicles.  
Used features: Vehicle Speed[km/h], Distance[km], Generalized\_Weight, MAF[g/sec], Absolute Load[%], Short Term Fuel Trim Bank 1[%], Short Term Fuel Trim Bank 2[%], Long Term Fuel 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]: 0.0035  
 Distance[km]: 0.0002  
 Generalized\_Weight: 0.0001  
 MAF[g/sec]: 0.0372  
 Absolute Load[%]: 0.0066  
 Short Term Fuel Trim Bank 1[%]: -0.0004  
 Short Term Fuel Trim Bank 2[%]: 0.0029  
 Long Term Fuel Trim Bank 1[%]: 0.0076  
 Long Term Fuel Trim Bank 2[%]: 0.0005  
  
Regression Equation:  
 Slope of the regression line: [ 3.51771144e-03 1.54859609e-04 8.60672719e-05 3.72308331e-02  
 6.57019976e-03 -3.68340265e-04 2.92205509e-03 7.63933480e-03  
 4.79373945e-04]  
 Intercept: -0.2424  
 Target Variable: FCR  
 FCR = 0.0035\*Vehicle Speed[km/h] + 0.0002\*Distance[km] + 0.0001\*Generalized\_Weight + 0.0372\*MAF[g/sec] + 0.0066\*Absolute Load[%] + -0.0004\*Short Term Fuel Trim Bank 1[%] + 0.0029\*Short Term Fuel Trim Bank 2[%] + 0.0076\*Long Term Fuel Trim Bank 1[%] + 0.0005\*Long Term Fuel Trim Bank 2[%] + -0.2424  
  
Evaluation Metrics:  
 R² Score : 0.6887  
 MAE : 0.0770  
 MSE : 0.0132  
 RMSE : 0.1150



"""  
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]: 0.0629  
  
Regression Equation:  
 Slope of the regression line: [0.06293091]  
 Intercept: 0.0413  
 Target Variable: FCR  
 FCR = 0.0629\*MAF[g/sec] + 0.0413  
  
Evaluation Metrics:  
 R² Score : 0.9440  
 MAE : 0.0216  
 MSE : 0.0013  
 RMSE : 0.0359



'''  
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()

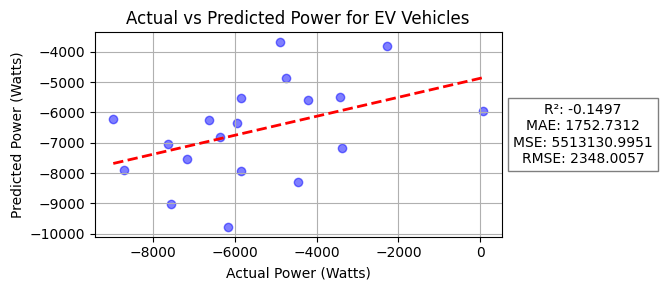
Trip Vehicle Type Latitude[deg] Longitude[deg] \  
822 554 EV 42.244071 -83.736644   
2172 1143 EV 42.244397 -83.731922   
842 565 EV 42.243102 -83.721478   
2250 1175 EV 42.254800 -83.725075   
1098 678 EV 42.272546 -83.749215   
  
 Air Conditioning Power[Watts] Heater Power[Watts] Vehicle Speed[km/h] \  
822 95.625000 182.812500 46.201749   
2172 0.000000 0.000000 54.838408   
842 0.000000 0.000000 51.888828   
2250 465.562249 489.457831 6.612898   
1098 0.000000 0.000000 31.160595   
  
 Distance[km] Engine RPM[RPM] OAT[DegC] ... MAF[g/sec] \  
822 0.881169 0.0 11.500000 ... 0.0   
2172 0.557468 0.0 15.000000 ... 0.0   
842 0.495307 0.0 10.000000 ... 0.0   
2250 0.212290 0.0 2.767068 ... 0.0   
1098 0.896625 0.0 15.072848 ... 0.0   
  
 Absolute Load[%] Short Term Fuel Trim Bank 1[%] \  
822 0.0 0.0   
2172 0.0 0.0   
842 0.0 0.0   
2250 0.0 0.0   
1098 0.0 0.0   
  
 Short Term Fuel Trim Bank 2[%] Long Term Fuel Trim Bank 1[%] \  
822 0.0 0.0   
2172 0.0 0.0   
842 0.0 0.0   
2250 0.0 0.0   
1098 0.0 0.0   
  
 Long Term Fuel Trim Bank 2[%] Air Conditioning Power[Watts]\_outlier \  
822 0.0 False   
2172 0.0 False   
842 0.0 False   
2250 0.0 False   
1098 0.0 False   
  
 Heater Power[Watts]\_outlier Vehicle Speed[km/h]\_outlier \  
822 False False   
2172 False False   
842 False False   
2250 False True   
1098 False False   
  
 HV Battery Power[Watts]\_outlier   
822 True   
2172 False   
842 False   
2250 False   
1098 False   
  
[5 rows x 23 columns]

df\_EV.describe()

Trip Latitude[deg] Longitude[deg] \  
count 95.000000 95.000000 95.000000   
mean 1274.105263 42.271297 -83.735505   
std 533.419428 0.017869 0.023455   
min 554.000000 42.231394 -83.792418   
25% 737.000000 42.260824 -83.749556   
50% 1149.000000 42.272546 -83.736166   
75% 1806.000000 42.284664 -83.723986   
max 2165.000000 42.307465 -83.682248   
  
 Air Conditioning Power[Watts] Heater Power[Watts] \  
count 95.000000 95.000000   
mean 451.206427 595.948098   
std 389.532228 766.175778   
min 0.000000 0.000000   
25% 79.973193 0.000000   
50% 370.610687 307.565789   
75% 785.391036 852.601621   
max 1224.153166 3545.282369   
  
 Vehicle Speed[km/h] Distance[km] Engine RPM[RPM] OAT[DegC] \  
count 95.000000 95.000000 95.0 95.000000   
mean 38.120463 2.529701 0.0 2.758068   
std 12.568234 2.356622 0.0 6.301469   
min 3.490542 0.174693 0.0 -13.153201   
25% 30.045382 0.879309 0.0 -2.118267   
50% 39.347546 1.704800 0.0 2.894870   
75% 45.667038 3.458791 0.0 7.279737   
max 67.570289 11.031075 0.0 15.156609   
  
 Generalized\_Weight FCR HV Battery Power[Watts] MAF[g/sec] \  
count 95.0 95.0 95.000000 95.0   
mean 3500.0 0.0 -6257.533768 0.0   
std 0.0 0.0 2555.379815 0.0   
min 3500.0 0.0 -13857.507261 0.0   
25% 3500.0 0.0 -7292.243492 0.0   
50% 3500.0 0.0 -6377.998333 0.0   
75% 3500.0 0.0 -4515.746667 0.0   
max 3500.0 0.0 73.835938 0.0   
  
 Absolute Load[%] Short Term Fuel Trim Bank 1[%] \  
count 95.0 95.0   
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 95.0 95.0   
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 95.0   
mean 0.0   
std 0.0   
min 0.0   
25% 0.0   
50% 0.0   
75% 0.0   
max 0.0

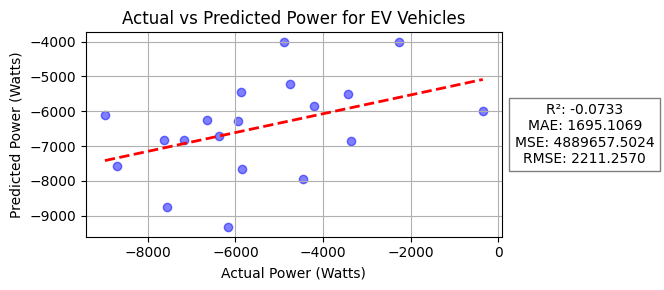
'''  
Scenario 1 for EV  
Selected features for linear regression analysis to predict HV Battery Power for EVs.  
Performed linear regression using the selected features and target.  
Plotted Actual vs Predicted Power for EV vehicles, including regression line and evaluation metrics (R², MAE, MSE, RMSE).  
'''  
  
'''  
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]: -2.5853  
 Heater Power[Watts]: -1.2364  
 Vehicle Speed[km/h]: -112.8938  
  
Regression Equation:  
 Slope of the regression line: [ -2.58530326 -1.23642892 -112.89382874]  
 Intercept: -271.2544  
 Target Variable: HV Battery Power[Watts]  
 HV Battery Power[Watts] = -2.5853\*Air Conditioning Power[Watts] + -1.2364\*Heater Power[Watts] + -112.8938\*Vehicle Speed[km/h] + -271.2544  
  
Evaluation Metrics:  
 R² Score : -0.1497  
 MAE : 1752.7312  
 MSE : 5513130.9951  
 RMSE : 2348.0057



'''  
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]: -2.0558  
 Heater Power[Watts]: -1.3454  
 Vehicle Speed[km/h]: -103.8481  
  
Regression Equation:  
 Slope of the regression line: [ -2.05582262 -1.34538495 -103.8481054 ]  
 Intercept: -764.9562  
 Target Variable: HV Battery Power[Watts]  
 HV Battery Power[Watts] = -2.0558\*Air Conditioning Power[Watts] + -1.3454\*Heater Power[Watts] + -103.8481\*Vehicle Speed[km/h] + -764.9562  
  
Evaluation Metrics:  
 R² Score : -0.0733  
 MAE : 1695.1069  
 MSE : 4889657.5024  
 RMSE : 2211.2570



df\_EV.describe()

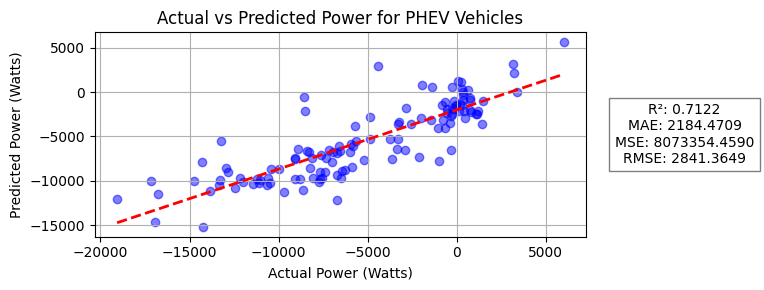
Trip Latitude[deg] Longitude[deg] \  
count 95.000000 95.000000 95.000000   
mean 1274.105263 42.271297 -83.735505   
std 533.419428 0.017869 0.023455   
min 554.000000 42.231394 -83.792418   
25% 737.000000 42.260824 -83.749556   
50% 1149.000000 42.272546 -83.736166   
75% 1806.000000 42.284664 -83.723986   
max 2165.000000 42.307465 -83.682248   
  
 Air Conditioning Power[Watts] Heater Power[Watts] \  
count 95.000000 95.000000   
mean 451.206427 595.948098   
std 389.532228 766.175778   
min 0.000000 0.000000   
25% 79.973193 0.000000   
50% 370.610687 307.565789   
75% 785.391036 852.601621   
max 1224.153166 3545.282369   
  
 Vehicle Speed[km/h] Distance[km] Engine RPM[RPM] OAT[DegC] \  
count 95.000000 95.000000 95.0 95.000000   
mean 38.120463 2.529701 0.0 2.758068   
std 12.568234 2.356622 0.0 6.301469   
min 3.490542 0.174693 0.0 -13.153201   
25% 30.045382 0.879309 0.0 -2.118267   
50% 39.347546 1.704800 0.0 2.894870   
75% 45.667038 3.458791 0.0 7.279737   
max 67.570289 11.031075 0.0 15.156609   
  
 Generalized\_Weight FCR HV Battery Power[Watts] MAF[g/sec] \  
count 95.0 95.0 95.000000 95.0   
mean 3500.0 0.0 -6257.533768 0.0   
std 0.0 0.0 2555.379815 0.0   
min 3500.0 0.0 -13857.507261 0.0   
25% 3500.0 0.0 -7292.243492 0.0   
50% 3500.0 0.0 -6377.998333 0.0   
75% 3500.0 0.0 -4515.746667 0.0   
max 3500.0 0.0 73.835938 0.0   
  
 Absolute Load[%] Short Term Fuel Trim Bank 1[%] \  
count 95.0 95.0   
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 95.0 95.0   
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 95.0   
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 618.000000 618.000000 618.000000   
mean 1027.393204 42.273889 -83.723610   
std 579.918702 0.019057 0.025063   
min 2.000000 42.229778 -83.799610   
25% 590.000000 42.259793 -83.739610   
50% 955.500000 42.277656 -83.719121   
75% 1457.750000 42.286126 -83.707332   
max 2432.000000 42.319032 -83.678096   
  
 Air Conditioning Power[Watts] Heater Power[Watts] \  
count 618.000000 618.0   
mean 27.315283 0.0   
std 165.769118 0.0   
min 0.000000 0.0   
25% 0.000000 0.0   
50% 0.000000 0.0   
75% 0.000000 0.0   
max 2008.721360 0.0   
  
 Vehicle Speed[km/h] Distance[km] Engine RPM[RPM] OAT[DegC] \  
count 618.000000 618.000000 618.000000 618.000000   
mean 42.794632 3.630850 452.272555 1.400766   
std 12.024760 3.956138 482.111349 6.943586   
min 14.977123 0.045077 0.000000 -20.775568   
25% 35.673644 1.383368 0.000000 -3.000000   
50% 41.934884 2.855270 346.214084 2.500000   
75% 47.761296 4.189252 783.824850 6.237949   
max 101.025354 51.932325 3360.000000 26.144342   
  
 Generalized\_Weight FCR HV Battery Power[Watts] MAF[g/sec] \  
count 618.000000 618.000000 618.000000 618.000000   
mean 3800.047327 1.310643 -5064.759049 4.287581   
std 366.069026 1.188460 5028.038070 3.193834   
min 3000.000000 0.000000 -23172.896785 0.170000   
25% 3806.372686 1.191592 -8178.594596 1.419221   
50% 4000.000000 1.310643 -5249.647151 4.287581   
75% 4000.000000 1.310643 -287.400366 5.717953   
max 4000.000000 7.870564 7689.712572 40.230000   
  
 Absolute Load[%] Short Term Fuel Trim Bank 1[%] \  
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   
  
 Short Term Fuel Trim Bank 2[%] Long Term Fuel Trim Bank 1[%] \  
count 618.0 0.0   
mean 0.0 NaN   
std 0.0 NaN   
min 0.0 NaN   
25% 0.0 NaN   
50% 0.0 NaN   
75% 0.0 NaN   
max 0.0 NaN   
  
 Long Term Fuel Trim Bank 2[%]   
count 618.0   
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()

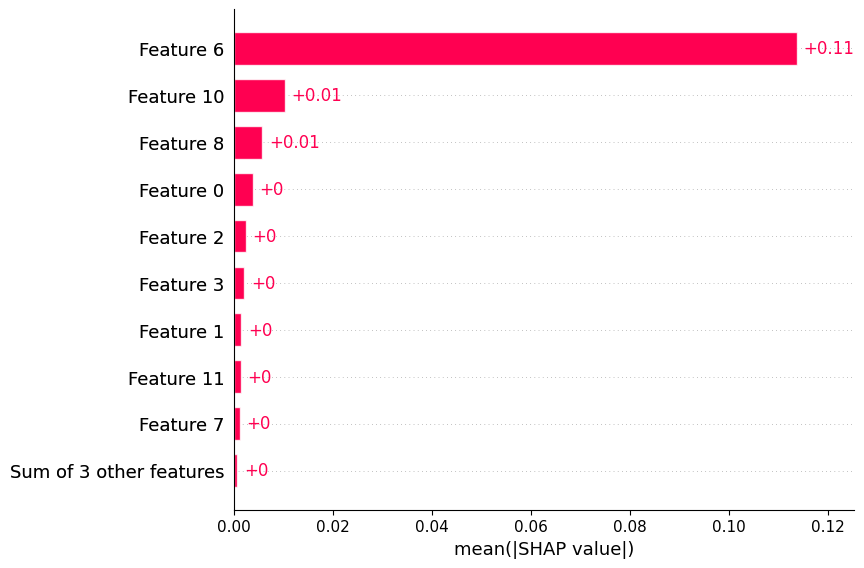
Model Coefficients:  
 Engine RPM[RPM]: 8.1661  
 Air Conditioning Power[Watts]: -0.9029  
 Vehicle Speed[km/h]: -203.6183  
 OAT[DegC]: 144.2664  
  
Regression Equation:  
 Slope of the regression line: [ 8.166065 -0.90289333 -203.61834751 144.26640201]  
 Intercept: -277.2092  
 Target Variable: HV Battery Power[Watts]  
 HV Battery Power[Watts] = 8.1661\*Engine RPM[RPM] + -0.9029\*Air Conditioning Power[Watts] + -203.6183\*Vehicle Speed[km/h] + 144.2664\*OAT[DegC] + -277.2092  
  
Evaluation Metrics:  
 R² Score : 0.7122  
 MAE : 2184.4709  
 MSE : 8073354.4590  
 RMSE : 2841.3649

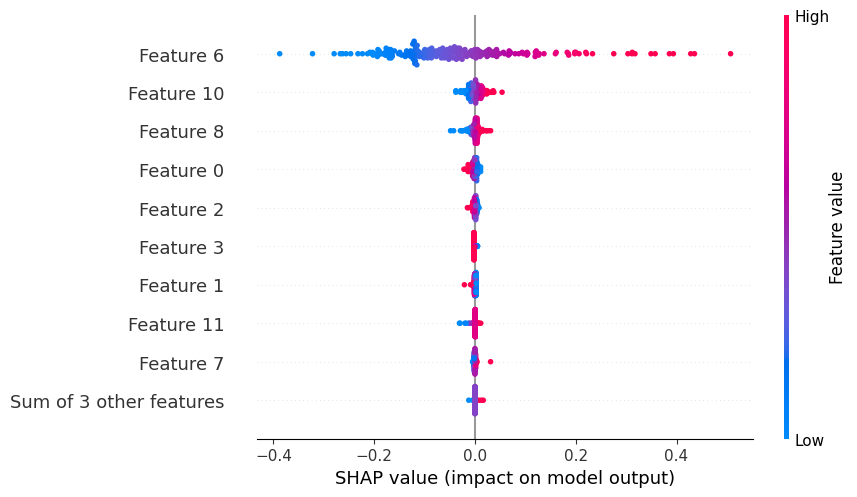


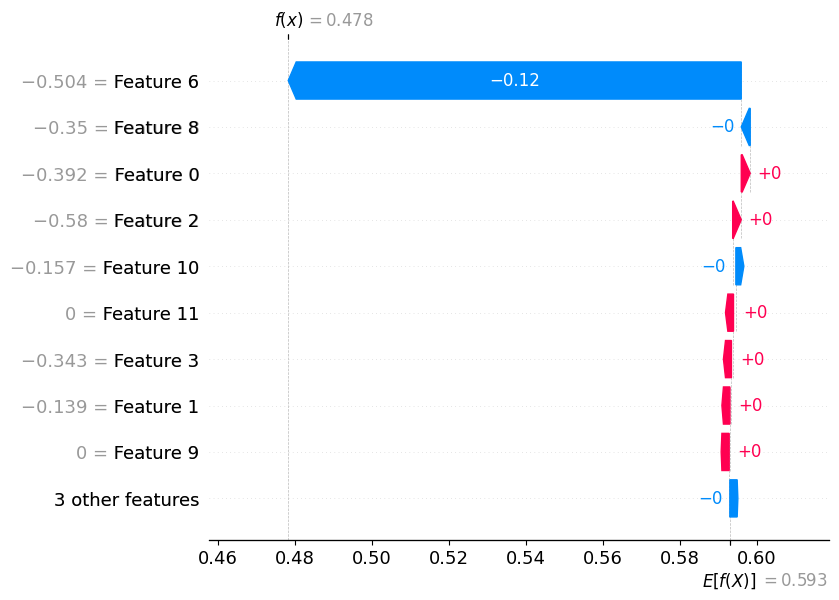
df\_ICE.info()

<class 'pandas.core.frame.DataFrame'>  
Index: 1850 entries, 3086 to 3723  
Data columns (total 19 columns):  
 # Column Non-Null Count Dtype   
--- ------ -------------- -----   
 0 Trip 1850 non-null int64   
 1 Vehicle Type 1850 non-null object   
 2 Latitude[deg] 1850 non-null float64  
 3 Longitude[deg] 1850 non-null float64  
 4 Air Conditioning Power[Watts] 1850 non-null float64  
 5 Heater Power[Watts] 0 non-null float64  
 6 Vehicle Speed[km/h] 1850 non-null float64  
 7 Distance[km] 1850 non-null float64  
 8 Engine RPM[RPM] 1850 non-null float64  
 9 OAT[DegC] 1850 non-null float64  
 10 Generalized\_Weight 1850 non-null float64  
 11 FCR 1850 non-null float64  
 12 HV Battery Power[Watts] 1850 non-null float64  
 13 MAF[g/sec] 1850 non-null float64  
 14 Absolute Load[%] 1850 non-null float64  
 15 Short Term Fuel Trim Bank 1[%] 1850 non-null float64  
 16 Short Term Fuel Trim Bank 2[%] 1850 non-null float64  
 17 Long Term Fuel Trim Bank 1[%] 1850 non-null float64  
 18 Long Term Fuel Trim Bank 2[%] 1850 non-null float64  
dtypes: float64(17), int64(1), object(1)  
memory usage: 289.1+ 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 0.002323 0.000396 0.002229 0.000534   
1 -0.004610 0.000477 0.000417 0.002119   
2 0.006633 -0.000043 -0.000759 -0.001930   
3 -0.001457 -0.000090 0.001661 0.000494   
4 0.000815 0.000157 0.000368 -0.001640   
  
 Generalized\_Weight HV Battery Power[Watts] MAF[g/sec] Absolute Load[%] \  
0 -0.000119 -0.0 -0.117554 -0.000057   
1 0.000116 -0.0 0.185491 0.000156   
2 -0.000119 -0.0 -0.109387 0.000487   
3 -0.000119 -0.0 -0.104673 -0.000157   
4 -0.000119 -0.0 -0.120312 -0.000818   
  
 Short Term Fuel Trim Bank 1[%] Short Term Fuel Trim Bank 2[%] \  
0 -0.002329 0.000192   
1 0.005444 0.000192   
2 -0.010127 0.000192   
3 0.003040 0.000192   
4 -0.000773 0.000192   
  
 Long Term Fuel Trim Bank 1[%] Long Term Fuel Trim Bank 2[%]   
0 -0.000825 0.000613   
1 -0.010098 0.000613   
2 -0.006545 0.000613   
3 0.015050 0.000613   
4 0.004881 0.000613

# 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)

push\_df\_to\_s3\_parquet(df\_combined, 'Cleaned up VED Source Data/df\_combined.parquet')

DataFrame uploaded to s3://s3aravindh973515031797/Cleaned up VED Source Data/df\_combined.parquet

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]']]

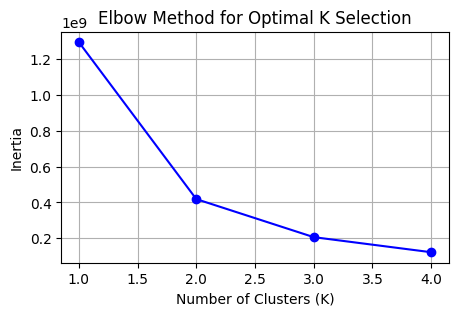
df\_combined\_of = df\_combined[['OAT[DegC]','FCR']]

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

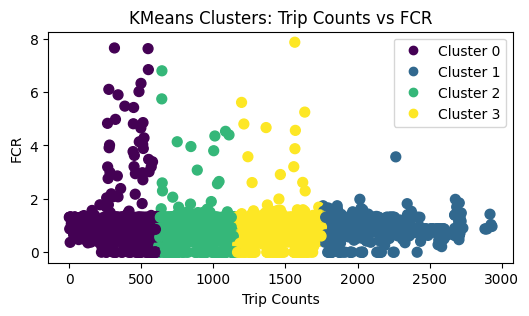


'''  
This cell:  
- Imports KMeans from sklearn.  
- Defines a function `fit\_predict\_kmeans` that:  
 - Fits KMeans clustering to the input DataFrame with a specified number of clusters.  
 - Predicts cluster labels for the input data.  
 - Adds a new 'Cluster' column to the DataFrame with the assigned 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)

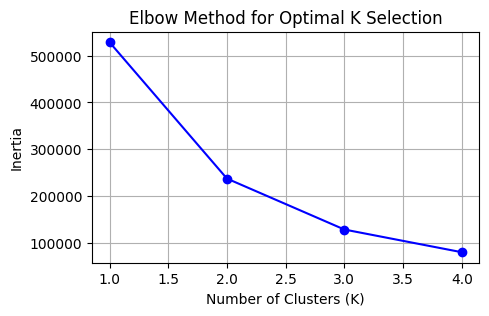
( Trip FCR Cluster  
 0 1637 2.298379 3  
 1 293 2.188477 0  
 2 340 2.076016 0  
 3 1455 1.994577 3  
 4 2016 1.984249 1  
 ... ... ... ...  
 3722 658 0.000000 2  
 3723 728 0.000000 2  
 3724 1674 0.000000 3  
 3725 1026 0.000000 2  
 3726 596 0.000000 0  
   
 [3727 rows x 3 columns],  
 array([3, 0, 0, ..., 3, 2, 0], shape=(3727,), 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)

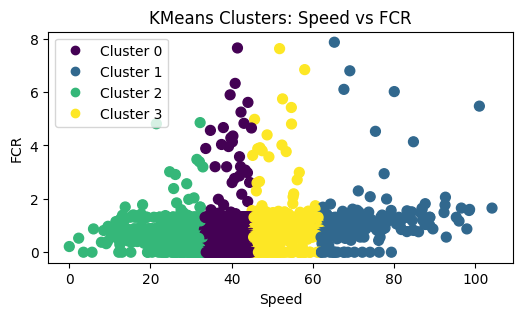
Suggested optimal K: 2



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

( Vehicle Speed[km/h] FCR Cluster  
 0 71.199588 2.298379 1  
 1 55.280435 2.188477 3  
 2 55.131579 2.076016 3  
 3 78.166852 1.994577 1  
 4 36.728477 1.984249 0  
 ... ... ... ...  
 3722 65.420500 0.000000 1  
 3723 64.908588 0.000000 1  
 3724 36.466653 0.000000 0  
 3725 58.251157 0.000000 3  
 3726 65.121893 0.000000 1  
   
 [3727 rows x 3 columns],  
 array([1, 3, 3, ..., 0, 3, 1], shape=(3727,), 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()



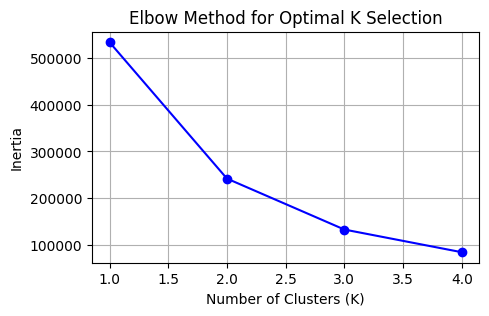
df\_combined\_sf = df\_combined[['Vehicle Type','Vehicle Speed[km/h]','FCR']]  
df\_combined\_sf.head()

Vehicle Type Vehicle Speed[km/h] FCR  
0 ICE 71.199588 2.298379  
1 ICE 55.280435 2.188477  
2 ICE 55.131579 2.076016  
3 ICE 78.166852 1.994577  
4 ICE 36.728477 1.984249

'''  
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)

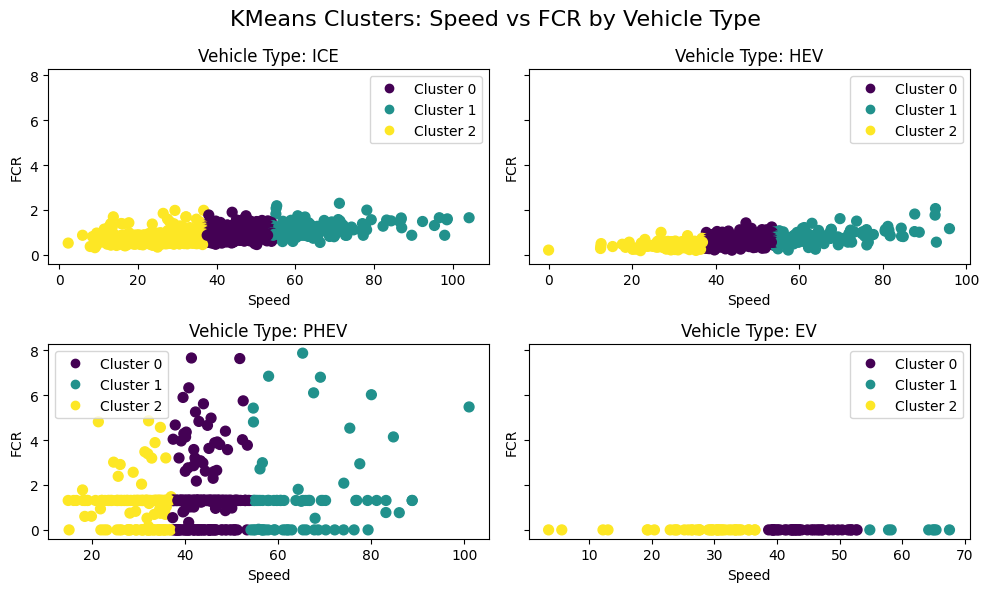
Suggested optimal K: 2



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

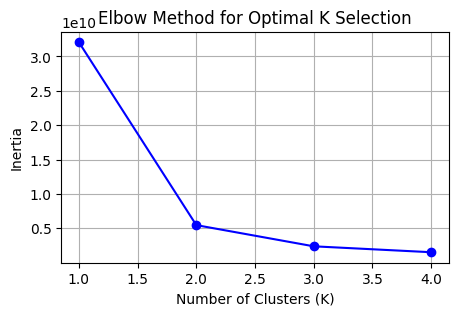
( Vehicle Type Vehicle Speed[km/h] FCR Cluster  
 0 0 71.199588 2.298379 1  
 1 0 55.280435 2.188477 1  
 2 0 55.131579 2.076016 1  
 3 0 78.166852 1.994577 1  
 4 0 36.728477 1.984249 2  
 ... ... ... ... ...  
 3722 2 65.420500 0.000000 1  
 3723 2 64.908588 0.000000 1  
 3724 2 36.466653 0.000000 2  
 3725 2 58.251157 0.000000 1  
 3726 2 65.121893 0.000000 1  
   
 [3727 rows x 4 columns],  
 array([1, 1, 1, ..., 2, 1, 1], shape=(3727,), 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)

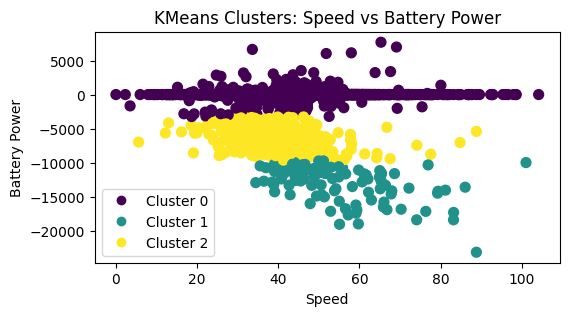
Suggested optimal K: 2



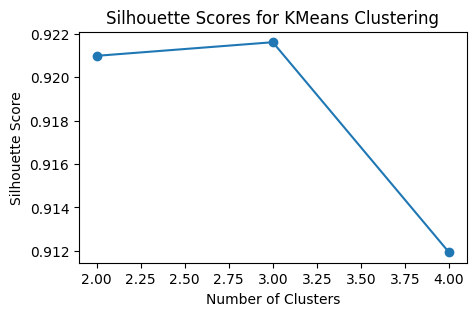
fit\_predict\_kmeans(df\_combined\_sb,3)

( Vehicle Speed[km/h] HV Battery Power[Watts] Cluster  
 0 71.199588 0.000000 0  
 1 55.280435 0.000000 0  
 2 55.131579 0.000000 0  
 3 78.166852 0.000000 0  
 4 36.728477 0.000000 0  
 ... ... ... ...  
 3722 65.420500 -11560.435733 1  
 3723 64.908588 -12142.036859 1  
 3724 36.466653 -12732.903288 1  
 3725 58.251157 -13526.524390 1  
 3726 65.121893 -13857.507261 1  
   
 [3727 rows x 3 columns],  
 array([0, 0, 0, ..., 1, 1, 1], shape=(3727,), 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.show()

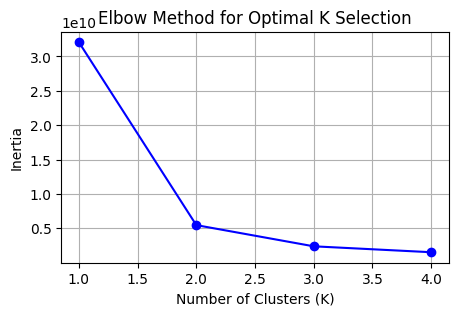


df\_combined\_sb = df\_combined[['Vehicle Type','Vehicle Speed[km/h]','HV Battery Power[Watts]']]  
# map vehicle type to 0,1,2,3  
df\_combined\_sb['Vehicle Type'] = df\_combined\_sb['Vehicle Type'].map({'ICE': 0, 'HEV': 1, 'EV': 2, 'PHEV': 3})  
  
df\_combined\_sb.head()

Vehicle Type Vehicle Speed[km/h] HV Battery Power[Watts]  
0 0 71.199588 0.0  
1 0 55.280435 0.0  
2 0 55.131579 0.0  
3 0 78.166852 0.0  
4 0 36.728477 0.0

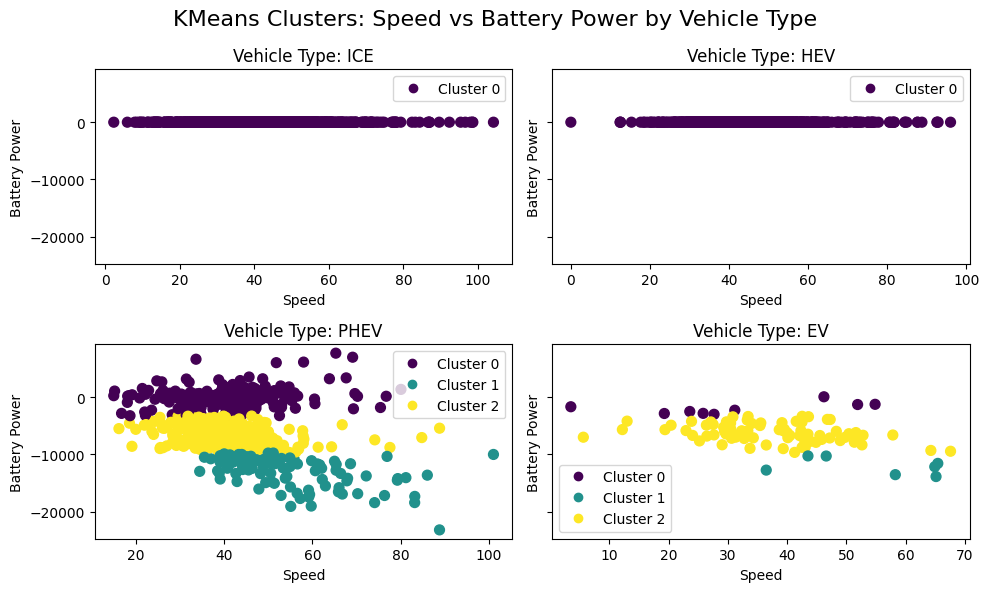
plot\_kmeans\_elbow(df\_combined\_sb)  
fit\_predict\_kmeans(df\_combined\_sb,3)

Suggested optimal K: 2



( Vehicle Type Vehicle Speed[km/h] HV Battery Power[Watts] Cluster  
 0 0 71.199588 0.000000 0  
 1 0 55.280435 0.000000 0  
 2 0 55.131579 0.000000 0  
 3 0 78.166852 0.000000 0  
 4 0 36.728477 0.000000 0  
 ... ... ... ... ...  
 3722 2 65.420500 -11560.435733 1  
 3723 2 64.908588 -12142.036859 1  
 3724 2 36.466653 -12732.903288 1  
 3725 2 58.251157 -13526.524390 1  
 3726 2 65.121893 -13857.507261 1  
   
 [3727 rows x 4 columns],  
 array([0, 0, 0, ..., 1, 1, 1], shape=(3727,), 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()

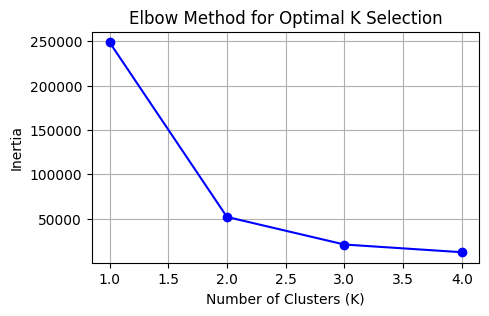


df\_combined\_sb

Vehicle Type Vehicle Speed[km/h] HV Battery Power[Watts] Cluster  
0 0 71.199588 0.000000 0  
1 0 55.280435 0.000000 0  
2 0 55.131579 0.000000 0  
3 0 78.166852 0.000000 0  
4 0 36.728477 0.000000 0  
... ... ... ... ...  
3722 2 65.420500 -11560.435733 1  
3723 2 64.908588 -12142.036859 1  
3724 2 36.466653 -12732.903288 1  
3725 2 58.251157 -13526.524390 1  
3726 2 65.121893 -13857.507261 1  
  
[3727 rows x 4 columns]

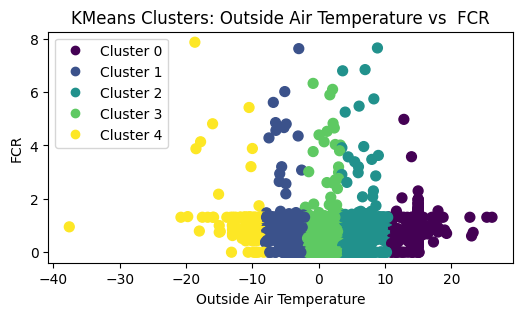
plot\_kmeans\_elbow(df\_combined\_of)  
fit\_predict\_kmeans(df\_combined\_of,5)

Suggested optimal K: 2



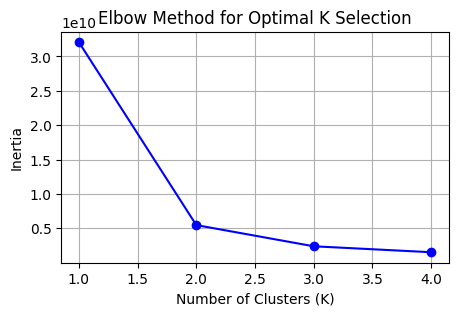
( OAT[DegC] FCR Cluster  
 0 15.000000 2.298379 0  
 1 -4.954545 2.188477 1  
 2 6.611650 2.076016 2  
 3 15.000000 1.994577 0  
 4 15.000000 1.984249 0  
 ... ... ... ...  
 3722 2.673522 0.000000 3  
 3723 6.500000 0.000000 2  
 3724 1.240812 0.000000 3  
 3725 -2.682927 0.000000 1  
 3726 -5.061386 0.000000 1  
   
 [3727 rows x 3 columns],  
 array([0, 1, 2, ..., 3, 1, 1], shape=(3727,), 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 -4.954545 0.000000 0  
 2 6.611650 0.000000 0  
 3 15.000000 0.000000 0  
 4 15.000000 0.000000 0  
 ... ... ... ...  
 3722 2.673522 -11560.435733 1  
 3723 6.500000 -12142.036859 1  
 3724 1.240812 -12732.903288 1  
 3725 -2.682927 -13526.524390 1  
 3726 -5.061386 -13857.507261 3  
   
 [3727 rows x 3 columns],  
 array([0, 0, 0, ..., 1, 1, 3], shape=(3727,), 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()

