This notebook will investigate the speeds for all entries.

In [10]:
```
# Python standard library imports
import time

# Third-party imports for database connection and data manipulation
from sqlalchemy import create_engine
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
# Third-party imports for mapping
import folium
import pandas as pd
from sqlalchemy import create_engine
from geopy.distance import geodesic
import numpy as np
```

Section 2: Connection

In [11]:
```
# Database connection parameters
dbname = 'DataMining'
user = 'postgres'
password = 'datamining'
host = 'localhost'  # localhost or the server address
port = '5433'  # default PostgreSQL port is 5432

# Establish a connection to the database
connection_str = f"postgresql://{user}:{password}@{host}:{port}/{dbname}"
engine = create_engine(connection_str)
```

Section 3: Define and Execute Query

In [12]:
```
sql_query = """
    SELECT DISTINCT mapped_veh_id
    FROM vehicle_data;
"""
vehicle_ids = pd.read_sql(sql_query, engine)['mapped_veh_id']
```

For each entry we calculate the speed. We filter out entries where the speed is higher than 200km/h because this is not possible. This might indicate a malfunctioning gps sensor or something going wrong in the data collection centre.

A video showcasing the problem was made

In [13]:
```
def calculate_distance(row):
    if pd.notna(row['prev_lat']) and pd.notna(row['prev_lon']):
        return geodesic((row['lat'], row['lon']), (row['prev_lat'], row['pre
    else:
        return None

def process_vehicle_data(veh_id):
    sql_query = f"""
        SELECT
            mapped_veh_id,
            timestamps_utc,
            lat,
            lon
        FROM
            vehicle_data
        WHERE
            mapped_veh_id = {veh_id}
        ORDER BY
            timestamps_utc;
    """
    df = pd.read_sql(sql_query, engine)

    # Calculate time difference in minutes
    df['prev_time'] = df.groupby('mapped_veh_id')['timestamps_utc'].shift(1
    df['time_diff_minutes'] = (df['timestamps_utc'] - df['prev_time']).dt.t

    # Calculate distance between consecutive points
    df['prev_lat'] = df.groupby('mapped_veh_id')['lat'].shift(1)
    df['prev_lon'] = df.groupby('mapped_veh_id')['lon'].shift(1)

    df['distance_km'] = df.apply(calculate_distance, axis=1)

    # Calculate speed in km/h
    df['speed_kmh'] = df['distance_km'] / (df['time_diff_minutes'] / 60)

    # Identify anomalies - considering distance greater than 1km and speed
    df['status'] = np.where((df['speed_kmh'] > 200) & (df['distance_km'] >

    anomalies['anom_id'] = 'locat_anom'
    # Return anomalies
    return df[df['status'] == 'Anomaly']
```

In [14]:

```
first_write = True
for veh_id in vehicle_ids:
    anomalies = process_vehicle_data(veh_id)

    if not anomalies.empty:
        if first_write:
            anomalies.to_csv('anomalies.csv', mode='w', header=True, index=
            first_write = False
        else:
            anomalies.to_csv('anomalies.csv', mode='a', header=False, index
```

```
/var/folders/30/bbv8h_y57kn0cxm37nn_rfx40000gn/T/ipykernel_46414/778744686.
py:39: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  anomalies['anom_id'] = 'locat_anom'
```