Exploratory Data Analysis

In [60]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
pip install wordnet
```

```
Requirement already satisfied: wordnet in d:\data_690\nlp\project\nlp_690\lib\site-pac
kages (0.0.1b2)
Collecting colorama==0.3.9 (from wordnet)
  Using cached colorama-0.3.9-py2.py3-none-any.whl (20 kB)
Installing collected packages: colorama
  Attempting uninstall: colorama
    Found existing installation: colorama 0.4.6
    Uninstalling colorama-0.4.6:
      Successfully uninstalled colorama-0.4.6
Successfully installed colorama-0.3.9
Note: you may need to restart the kernel to use updated packages.
```

```
ERROR: pip's dependency resolver does not currently take into account all the packages
that are installed. This behaviour is the source of the following dependency conflict
s.
wasabi 1.1.2 requires colorama>=0.4.6; sys_platform == "win32" and python_version >=
"3.7", but you have colorama 0.3.9 which is incompatible.
```

In [4]:
```
pip install spacy
```

```
Requirement already satisfied: spacy in d:\data_690\nlp\project\nlp_690\lib\site-packa
ges (3.7.2)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in d:\data_690\nlp\project
\nlp_690\lib\site-packages (from spacy) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in d:\data_690\nlp\project
\nlp_690\lib\site-packages (from spacy) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in d:\data_690\nlp\project\nl
p_690\lib\site-packages (from spacy) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from spacy) (2.0.8)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in d:\data_690\nlp\project\nlp_69
0\lib\site-packages (from spacy) (3.0.9)
Requirement already satisfied: thinc<8.3.0,>=8.1.8 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from spacy) (8.2.1)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from spacy) (1.1.2)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from spacy) (2.4.8)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in d:\data_690\nlp\project\nlp_
690\lib\site-packages (from spacy) (2.0.10)
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from spacy) (0.3.4)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from spacy) (0.9.0)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in d:\data_690\nlp\project\nlp
_690\lib\site-packages (from spacy) (6.4.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from spacy) (4.66.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in d:\data_690\nlp\project\nlp_
690\lib\site-packages (from spacy) (2.31.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in d:\data_690\nlp
\project\nlp_690\lib\site-packages (from spacy) (2.5.2)
Requirement already satisfied: jinja2 in d:\data_690\nlp\project\nlp_690\lib\site-pack
ages (from spacy) (3.1.2)
Requirement already satisfied: setuptools in d:\data_690\nlp\project\nlp_690\lib\site-
packages (from spacy) (68.2.2)
Requirement already satisfied: packaging>=20.0 in d:\data_690\nlp\project\nlp_690\lib
\site-packages (from spacy) (23.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in d:\data_690\nlp\project\nlp_
690\lib\site-packages (from spacy) (3.3.0)
Requirement already satisfied: numpy>=1.19.0 in d:\data_690\nlp\project\nlp_690\lib\si
te-packages (from spacy) (1.26.1)
Requirement already satisfied: annotated-types>=0.4.0 in d:\data_690\nlp\project\nlp_6
90\lib\site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (0.6.0)
Requirement already satisfied: pydantic-core==2.14.5 in d:\data_690\nlp\project\nlp_69
0\lib\site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (2.14.5)
Requirement already satisfied: typing-extensions>=4.6.1 in d:\data_690\nlp\project\nlp
_690\lib\site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy) (4.8.0)
Requirement already satisfied: charset-normalizer<4,>=2 in d:\data_690\nlp\project\nlp
_690\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (3.3.0)
Requirement already satisfied: idna<4,>=2.5 in d:\data_690\nlp\project\nlp_690\lib\sit
e-packages (from requests<3.0.0,>=2.13.0->spacy) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in d:\data_690\nlp\project\nlp_690\l
ib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (2.0.6)
Requirement already satisfied: certifi>=2017.4.17 in d:\data_690\nlp\project\nlp_690\l
ib\site-packages (from requests<3.0.0,>=2.13.0->spacy) (2023.7.22)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in d:\data_690\nlp\project\nlp_690\l
ib\site-packages (from thinc<8.3.0,>=8.1.8->spacy) (0.7.11)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in d:\data_690\nlp\project\nlp
_690\lib\site-packages (from thinc<8.3.0,>=8.1.8->spacy) (0.1.4)
```

```
Requirement already satisfied: colorama in d:\data_690\nlp\project\nlp_690\lib\site-packages (from tqdm<5.0.0,>=4.38.0->spacy) (0.3.9)
Requirement already satisfied: click<9.0.0,>=7.1.1 in d:\data_690\nlp\project\nlp_690\lib\site-packages (from typer<0.10.0,>=0.3.0->spacy) (8.1.7)
Collecting colorama (from tqdm<5.0.0,>=4.38.0->spacy)
  Using cached colorama-0.4.6-py2.py3-none-any.whl (25 kB)
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in d:\data_690\nlp\project\nlp_690\lib\site-packages (from weasel<0.4.0,>=0.1.0->spacy) (0.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in d:\data_690\nlp\project\nlp_690\lib\site-packages (from jinja2->spacy) (2.1.3)
Installing collected packages: colorama
  Attempting uninstall: colorama
    Found existing installation: colorama 0.3.9
    Uninstalling colorama-0.3.9:
      Successfully uninstalled colorama-0.3.9
Successfully installed colorama-0.4.6
Note: you may need to restart the kernel to use updated packages.
```

```
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source of the following dependency conflicts.
wordnet 0.0.1b2 requires colorama==0.3.9, but you have colorama 0.4.6 which is incompatible.
```

In [5]:

```
pip install plotly
```

```
Requirement already satisfied: plotly in d:\data_690\nlp\project\nlp_690\lib\site-packages (5.18.0)
Requirement already satisfied: tenacity>=6.2.0 in d:\data_690\nlp\project\nlp_690\lib\site-packages (from plotly) (8.2.3)
Requirement already satisfied: packaging in d:\data_690\nlp\project\nlp_690\lib\site-packages (from plotly) (23.2)
Note: you may need to restart the kernel to use updated packages.
```

In [6]:

```
!python -m spacy download en_core_web_sm
```

```
Collecting en-core-web-sm==3.7.1
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_
sm-3.7.1/en_core_web_sm-3.7.1-py3-none-any.whl (12.8 MB)
     ---------------------------------------- 0.0/12.8 MB ? eta -:--:--
     --------------------------------------- 0.0/12.8 MB 330.3 kB/s eta 0:00:39
     --------------------------------------- 0.1/12.8 MB 550.5 kB/s eta 0:00:24
      --------------------------------------- 0.2/12.8 MB 1.6 MB/s eta 0:00:08
     --- ------------------------------------ 1.0/12.8 MB 5.3 MB/s eta 0:00:03
     --------- ------------------------------ 3.0/12.8 MB 12.6 MB/s eta 0:00:01
     ------------ --------------------------- 4.2/12.8 MB 16.6 MB/s eta 0:00:01
     ------------------ --------------------- 5.8/12.8 MB 17.5 MB/s eta 0:00:01
     ------------------------- -------------- 8.0/12.8 MB 22.3 MB/s eta 0:00:01
     ------------------------- -------------- 8.0/12.8 MB 22.3 MB/s eta 0:00:01
     ------------------------------- ------- 10.3/12.8 MB 28.5 MB/s eta 0:00:01
     ------------------------------------ -- 11.9/12.8 MB 32.8 MB/s eta 0:00:01
     --------------------------------------  12.8/12.8 MB 29.7 MB/s eta 0:00:01
     --------------------------------------- 12.8/12.8 MB 28.4 MB/s eta 0:00:00
Requirement already satisfied: spacy<3.8.0,>=3.7.2 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from en-core-web-sm==3.7.1) (3.7.2)
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.11 in d:\data_690\nlp\project
\nlp_690\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.0.12)
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in d:\data_690\nlp\project
\nlp_690\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.0.5)
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in d:\data_690\nlp\project\nl
p_690\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.0.10)
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.0.8)
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in d:\data_690\nlp\project\nlp_69
0\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.0.9)
Requirement already satisfied: thinc<8.3.0,>=8.1.8 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (8.2.1)
Requirement already satisfied: wasabi<1.2.0,>=0.9.1 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.1.2)
Requirement already satisfied: srsly<3.0.0,>=2.4.3 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.4.8)
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in d:\data_690\nlp\project\nlp_
690\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.0.10)
Requirement already satisfied: weasel<0.4.0,>=0.1.0 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.3.4)
Requirement already satisfied: typer<0.10.0,>=0.3.0 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.9.0)
Requirement already satisfied: smart-open<7.0.0,>=5.2.1 in d:\data_690\nlp\project\nlp
_690\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (6.4.0)
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (4.66.1)
Requirement already satisfied: requests<3.0.0,>=2.13.0 in d:\data_690\nlp\project\nlp_
690\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.31.0)
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4 in d:\data_690\nlp
\project\nlp_690\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1)
(2.5.2)
Requirement already satisfied: jinja2 in d:\data_690\nlp\project\nlp_690\lib\site-pack
ages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.1.2)
Requirement already satisfied: setuptools in d:\data_690\nlp\project\nlp_690\lib\site-
packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (68.2.2)
Requirement already satisfied: packaging>=20.0 in d:\data_690\nlp\project\nlp_690\lib
\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (23.2)
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in d:\data_690\nlp\project\nlp_
690\lib\site-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (3.3.0)
Requirement already satisfied: numpy>=1.19.0 in d:\data_690\nlp\project\nlp_690\lib\si
te-packages (from spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (1.26.1)
```

```
Requirement already satisfied: annotated-types>=0.4.0 in d:\data_690\nlp\project\nlp_6
90\lib\site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->
en-core-web-sm==3.7.1) (0.6.0)
Requirement already satisfied: pydantic-core==2.14.5 in d:\data_690\nlp\project\nlp_69
0\lib\site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2->e
n-core-web-sm==3.7.1) (2.14.5)
Requirement already satisfied: typing-extensions>=4.6.1 in d:\data_690\nlp\project\nlp
_690\lib\site-packages (from pydantic!=1.8,!=1.8.1,<3.0.0,>=1.7.4->spacy<3.8.0,>=3.7.2
->en-core-web-sm==3.7.1) (4.8.0)
Requirement already satisfied: charset-normalizer<4,>=2 in d:\data_690\nlp\project\nlp
_690\lib\site-packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web
-sm==3.7.1) (3.3.0)
Requirement already satisfied: idna<4,>=2.5 in d:\data_690\nlp\project\nlp_690\lib\sit
e-packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1)
(3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in d:\data_690\nlp\project\nlp_690\l
ib\site-packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==
3.7.1) (2.0.6)
Requirement already satisfied: certifi>=2017.4.17 in d:\data_690\nlp\project\nlp_690\l
ib\site-packages (from requests<3.0.0,>=2.13.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==
3.7.1) (2023.7.22)
Requirement already satisfied: blis<0.8.0,>=0.7.8 in d:\data_690\nlp\project\nlp_690\l
ib\site-packages (from thinc<8.3.0,>=8.1.8->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.
1) (0.7.11)
Requirement already satisfied: confection<1.0.0,>=0.0.1 in d:\data_690\nlp\project\nlp
_690\lib\site-packages (from thinc<8.3.0,>=8.1.8->spacy<3.8.0,>=3.7.2->en-core-web-sm=
=3.7.1) (0.1.4)
Requirement already satisfied: colorama in d:\data_690\nlp\project\nlp_690\lib\site-pa
ckages (from tqdm<5.0.0,>=4.38.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (0.4.6)
Requirement already satisfied: click<9.0.0,>=7.1.1 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from typer<0.10.0,>=0.3.0->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.
7.1) (8.1.7)
Requirement already satisfied: cloudpathlib<0.17.0,>=0.7.0 in d:\data_690\nlp\project
\nlp_690\lib\site-packages (from weasel<0.4.0,>=0.1.0->spacy<3.8.0,>=3.7.2->en-core-we
b-sm==3.7.1) (0.16.0)
Requirement already satisfied: MarkupSafe>=2.0 in d:\data_690\nlp\project\nlp_690\lib
\site-packages (from jinja2->spacy<3.8.0,>=3.7.2->en-core-web-sm==3.7.1) (2.1.3)
[+] Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
```

In [7]:

```
pip install folium
```

```
Requirement already satisfied: folium in d:\data_690\nlp\project\nlp_690\lib\site-pack
ages (0.15.0)
Requirement already satisfied: branca>=0.6.0 in d:\data_690\nlp\project\nlp_690\lib\si
te-packages (from folium) (0.7.0)
Requirement already satisfied: jinja2>=2.9 in d:\data_690\nlp\project\nlp_690\lib\site
-packages (from folium) (3.1.2)
Requirement already satisfied: numpy in d:\data_690\nlp\project\nlp_690\lib\site-packa
ges (from folium) (1.26.1)
Requirement already satisfied: requests in d:\data_690\nlp\project\nlp_690\lib\site-pa
ckages (from folium) (2.31.0)
Requirement already satisfied: MarkupSafe>=2.0 in d:\data_690\nlp\project\nlp_690\lib
\site-packages (from jinja2>=2.9->folium) (2.1.3)
Requirement already satisfied: charset-normalizer<4,>=2 in d:\data_690\nlp\project\nlp
_690\lib\site-packages (from requests->folium) (3.3.0)
Requirement already satisfied: idna<4,>=2.5 in d:\data_690\nlp\project\nlp_690\lib\sit
e-packages (from requests->folium) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in d:\data_690\nlp\project\nlp_690\l
ib\site-packages (from requests->folium) (2.0.6)
Requirement already satisfied: certifi>=2017.4.17 in d:\data_690\nlp\project\nlp_690\l
ib\site-packages (from requests->folium) (2023.7.22)
Note: you may need to restart the kernel to use updated packages.
```

In [8]:
```
pip install geopy
```

```
Requirement already satisfied: geopy in d:\data_690\nlp\project\nlp_690\lib\site-packa
ges (2.4.1)
Requirement already satisfied: geographiclib<3,>=1.52 in d:\data_690\nlp\project\nlp_6
90\lib\site-packages (from geopy) (2.0)
Note: you may need to restart the kernel to use updated packages.
```

In [9]:
```
pip install WordCloud
```

```
Requirement already satisfied: WordCloud in d:\data_690\nlp\project\nlp_690\lib\site-p
ackages (1.9.2)
Requirement already satisfied: numpy>=1.6.1 in d:\data_690\nlp\project\nlp_690\lib\sit
e-packages (from WordCloud) (1.26.1)
Requirement already satisfied: pillow in d:\data_690\nlp\project\nlp_690\lib\site-pack
ages (from WordCloud) (10.1.0)
Requirement already satisfied: matplotlib in d:\data_690\nlp\project\nlp_690\lib\site-
packages (from WordCloud) (3.8.1)
Requirement already satisfied: contourpy>=1.0.1 in d:\data_690\nlp\project\nlp_690\lib
\site-packages (from matplotlib->WordCloud) (1.2.0)
Requirement already satisfied: cycler>=0.10 in d:\data_690\nlp\project\nlp_690\lib\sit
e-packages (from matplotlib->WordCloud) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in d:\data_690\nlp\project\nlp_690\li
b\site-packages (from matplotlib->WordCloud) (4.44.0)
Requirement already satisfied: kiwisolver>=1.3.1 in d:\data_690\nlp\project\nlp_690\li
b\site-packages (from matplotlib->WordCloud) (1.4.5)
Requirement already satisfied: packaging>=20.0 in d:\data_690\nlp\project\nlp_690\lib
\site-packages (from matplotlib->WordCloud) (23.2)
Requirement already satisfied: pyparsing>=2.3.1 in d:\data_690\nlp\project\nlp_690\lib
\site-packages (from matplotlib->WordCloud) (3.1.1)
Requirement already satisfied: python-dateutil>=2.7 in d:\data_690\nlp\project\nlp_690
\lib\site-packages (from matplotlib->WordCloud) (2.8.2)
Requirement already satisfied: importlib-resources>=3.2.0 in d:\data_690\nlp\project\n
lp_690\lib\site-packages (from matplotlib->WordCloud) (6.1.1)
Requirement already satisfied: zipp>=3.1.0 in d:\data_690\nlp\project\nlp_690\lib\site
-packages (from importlib-resources>=3.2.0->matplotlib->WordCloud) (3.17.0)
Requirement already satisfied: six>=1.5 in d:\data_690\nlp\project\nlp_690\lib\site-pa
ckages (from python-dateutil>=2.7->matplotlib->WordCloud) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [61]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import spacy
from gensim.corpora import Dictionary
from gensim.models import LdaModel
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
# from google.colab import drive
from sklearn.feature_extraction.text import CountVectorizer
import folium
from geopy.geocoders import Nominatim
from IPython.display import display
from wordcloud import WordCloud
```

In [62]:

```
import networkx as nx
import plotly.express as px
```

In [5]:

```
# Download NLTK resources
import nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('wordnet')
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\vanam\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\vanam\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\vanam\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[5]:
```
True
```

In [64]:
```
df_wc = pd.read_csv('dataset/final_cleaned_data/west_coast_cleaned_data.csv')
df_ec = pd.read_csv('dataset/final_cleaned_data/east_coast_cleaned_data.csv')
df_central = pd.read_csv('dataset/final_cleaned_data/central_cleaned_data.csv')
```

In [65]:
```
df = pd.concat([df_wc, df_ec, df_central], ignore_index=True)
df.describe()
```

Out[65]:

	Unnamed: 0
count	45.000000
mean	7.000000
std	4.369314
min	0.000000
25%	3.000000
50%	7.000000
75%	11.000000
max	14.000000

DataFrame Structure

The DataFrame (`df`) contains the following columns:

1. **'title':**
   - Represents the title of the news article.
2. **'article':**
   - Contains the body or content of the news article.
3. **'news_source':**
   - Indicates the name of the publishing agency or news source.
4. **'region':**
   - Specifies the region to which the news article belongs ('east-coast', 'central', 'west-coast').
5. **'article_cleaned':**

- Contains the preprocessed and cleaned version of the article text.

6. **'converted_date':**

   - Represents the date of the article, potentially in a converted format.

7. **'year':**

   - Represents the year of the article.

8. **'entities':**

   - Contains information about entities extracted from the article. This could include named entities such as people, organizations, locations, etc.

In [66]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 45 entries, 0 to 44
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Unnamed: 0       45 non-null     int64 
 1   title            45 non-null     object
 2   article          45 non-null     object
 3   news_source      45 non-null     object
 4   region           45 non-null     object
 5   article_cleaned  45 non-null     object
 6   converted_date   45 non-null     object
dtypes: int64(1), object(6)
memory usage: 2.6+ KB
```

In [67]:
```
df.head(1)
```

Out[67]:

	Unnamed: 0	title	article	news_source	region	article_cleaned	converted_date
0	0	Commentary: Driving an EV does not make you p...	['When I started driving an electric vehicle i...	latimes	west-coast	started driving electric vehicle 2018 became p...	09-17-2022

In [68]:
```
df['Unnamed: 0']
```

Out[68]:
```
0      0
1      1
2      2
3      3
4      4
5      5
6      6
7      7
8      8
9      9
10    10
11    11
12    12
13    13
14    14
15     0
16     1
17     2
18     3
19     4
20     5
21     6
22     7
23     8
24     9
25    10
26    11
27    12
28    13
29    14
30     0
31     1
32     2
33     3
34     4
35     5
36     6
37     7
38     8
39     9
40    10
41    11
42    12
43    13
44    14
Name: Unnamed: 0, dtype: int64
```

In [69]:
```
df.drop('Unnamed: 0', axis=1, inplace=True)
```

In [70]:
```
df.columns
```

Out[70]:
```
Index(['title', 'article', 'news_source', 'region', 'article_cleaned',
       'converted_date'],
      dtype='object')
```

In [71]:
```
df.index
```

Out[71]:
```
RangeIndex(start=0, stop=45, step=1)
```

In [72]:
```
df = df.rename_axis('article_no')
```

In [73]:
```
df.columns
```

Out[73]:
```
Index(['title', 'article', 'news_source', 'region', 'article_cleaned',
       'converted_date'],
      dtype='object')
```

In [74]:
```
df.head(1)
```

Out[74]:

article_no	title	article	news_source	region	article_cleaned	converted_date
0	Commentary: Driving an EV does not make you p...	['When I started driving an electric vehicle i...	latimes	west-coast	started driving electric vehicle 2018 became p...	09-17-2022

In [75]:
```
df.index
```

Out[75]:
```
RangeIndex(start=0, stop=45, step=1, name='article_no')
```

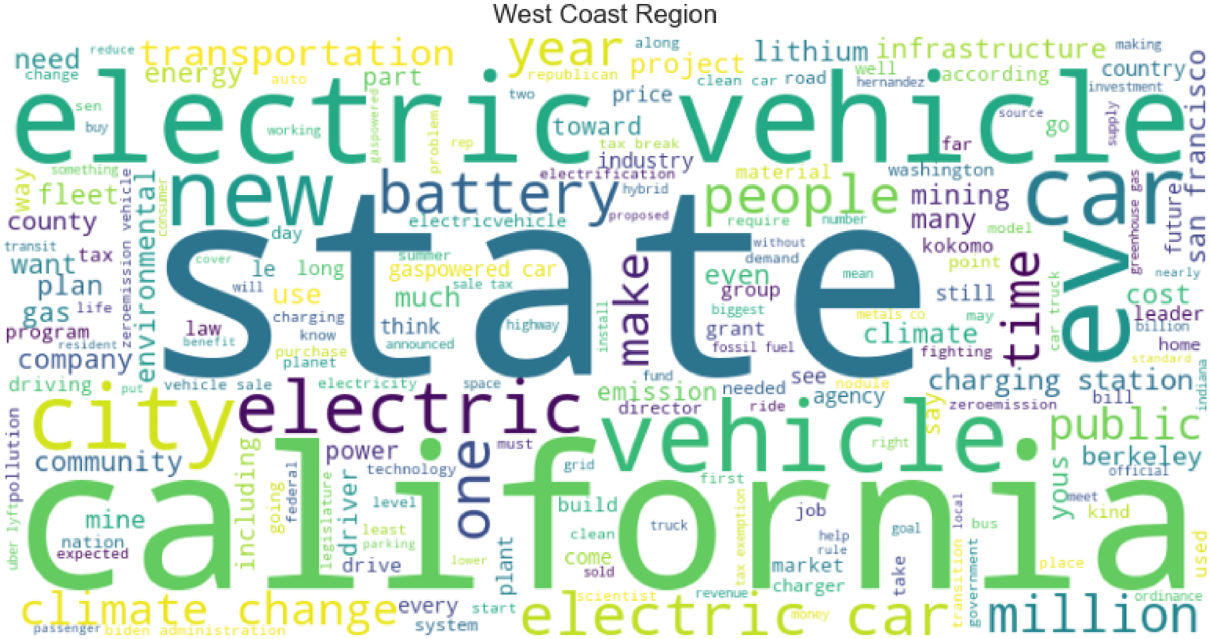
Wordclouds for each Region

In [76]:
```
# word cloud for Central
combined_text = ' '.join(df_central['article_cleaned'])

# Create a WordCloud object
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(combin

# Display the Word Cloud
plt.figure(figsize=(10, 5))
plt.title("Central Region")
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

Central Region

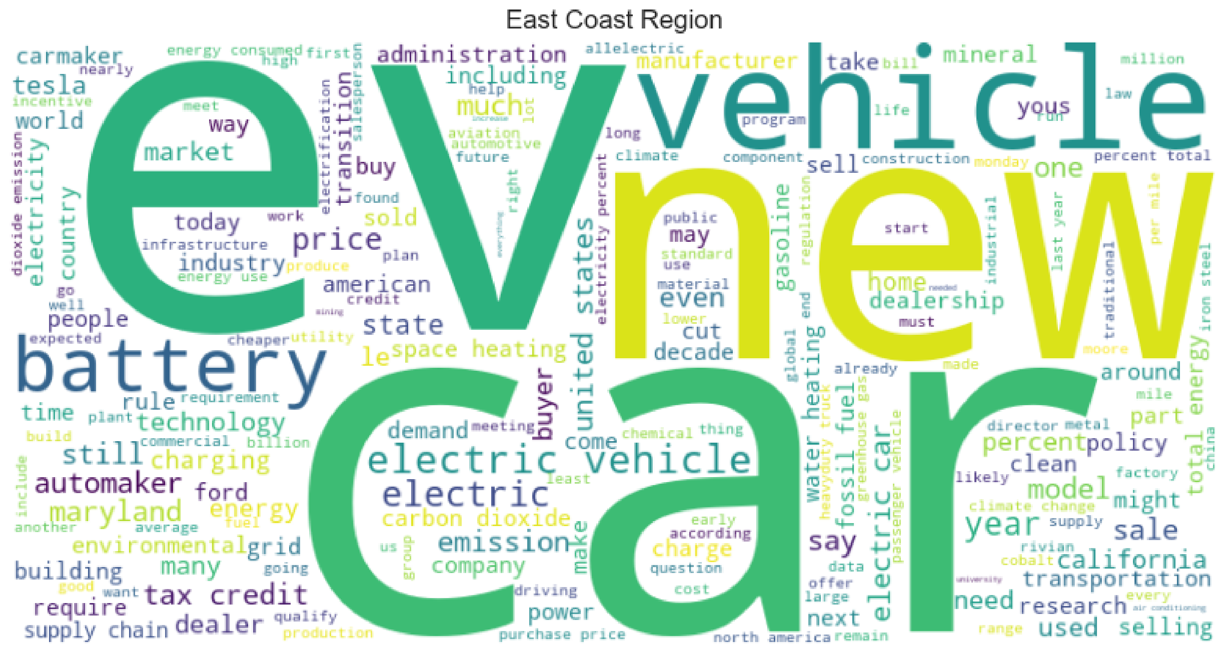
In [77]:
```
# word cloud for west coast
combined_text = ' '.join(df_wc['article_cleaned'])

# Create a WordCloud object
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(combin

# Display the Word Cloud
plt.figure(figsize=(10, 5))
plt.title("West Coast Region")
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```

West Coast Region



In [78]:
```
# word cloud for east coast
combined_text = ' '.join(df_ec['article_cleaned'])

# Create a WordCloud object
```

```python
wordcloud = WordCloud(width=800, height=400, background_color='white').generate(combin

# Display the Word Cloud
plt.figure(figsize=(10, 5))
plt.title("East Coast Region")
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
```




```python
In [79]: df['year'] = df['converted_date'].str.split('-').str[-1].astype(int)
```

```python
In [80]: df.head()
```

Out[80]:

article_no	title	article	news_source	region	article_cleaned	converted_date	y
0	Commentary: Driving an EV does not make you p...	['When I started driving an electric vehicle i...	latimes	west-coast	started driving electric vehicle 2018 became p...	09-17-2022	2(
1	Op-Ed: Think bigger. Switching to electric ca...	['It might feel like the easy solution — just ...	latimes	west-coast	might feel like easy solution replace gasguzzl...	09-15-2022	2(
2	Editorial: EPA wants to speed up EV switch. G...	['The Biden administration just proposed hitti...	latimes	west-coast	biden administration proposed hitting accelera...	04-12-2023	2(
3	California's electric car revolution, designe...	['The precious cargo on the ship docked in San...	latimes	west-coast	precious cargo ship docked san diego bay strik...	07-21-2021	2(
4	Electric cars now make up a fifth of Californ...	['One out of every 5 cars sold in California i...	latimes	west-coast	one every 5 car sold california powered batter...	11-01-2023	2(

In [81]:

```
print("Number of words per sample: " ,np.mean(df['article_cleaned'].apply(lambda x: le
```

```
Number of words per sample:  628.6666666666666
```

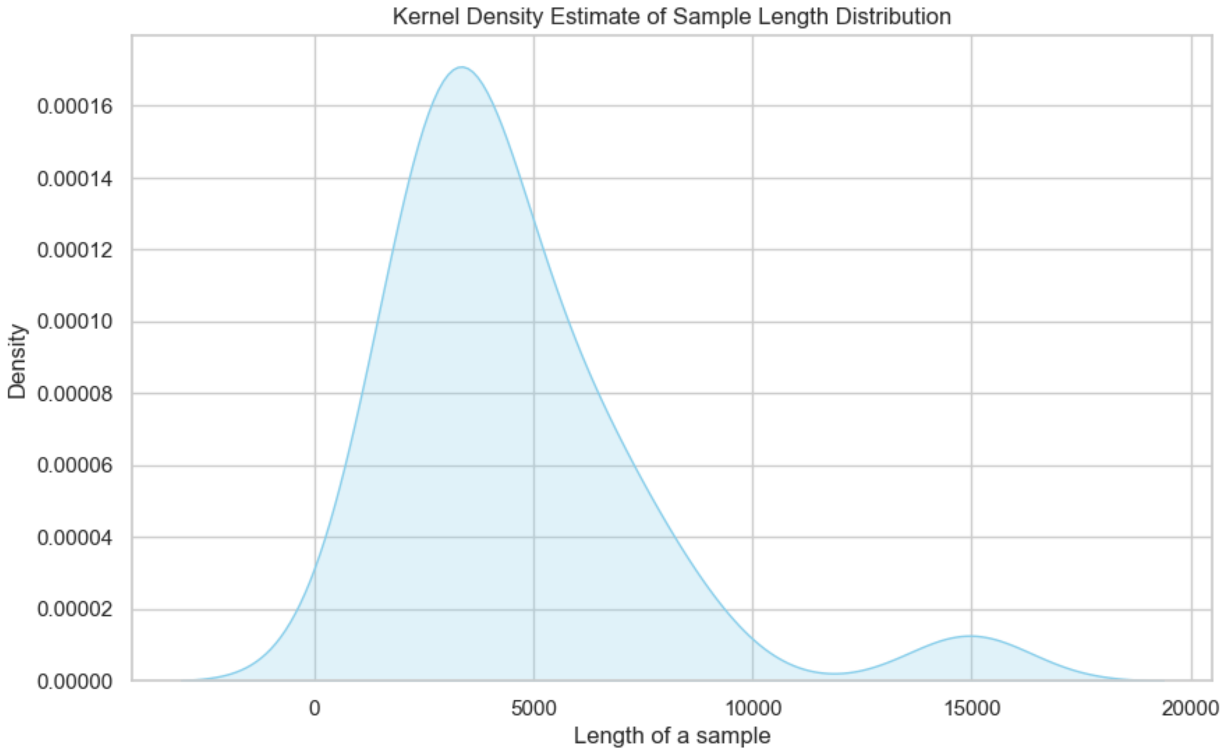
The code below (plot_sample_length_distribution function) generates and displays a KDE plot showing the distribution of the lengths of samples in the 'article' column of the DataFrame df. The x-axis represents the length of a sample, the y-axis represents the density, and the plot provides insights into how the lengths of the samples are distributed.

In [82]:

```
def plot_sample_length_distribution(sample_texts):
    """
        samples_texts
    """
    plt.figure(figsize=(10, 6))
    sns.kdeplot([len(s) for s in sample_texts], fill=True, color='skyblue')
    plt.xlabel('Length of a sample')
    plt.ylabel('Density')
    plt.title('Kernel Density Estimate of Sample Length Distribution')
    plt.show()
```

In [83]:

```
plot_sample_length_distribution(df['article_cleaned'].tolist())
```



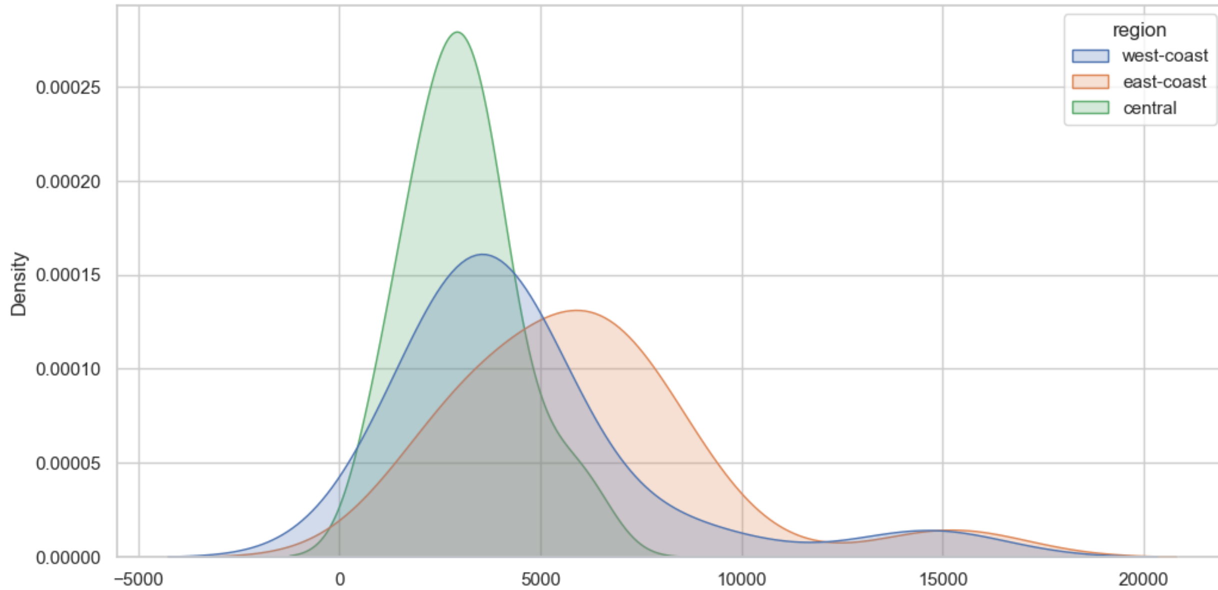
It's important to note that the values on the y-axis are not direct probabilities but rather a measure of how densely the data is distributed along the x-axis.Higher values on the y-axis indicate areas where the data is more densely distributed, and lower values indicate less dense regions. Based on the above distribution, we can infer that one or two articles were very long while the remaining articles were mostly of the range of similar lengths.

In [84]:
```
result = df.groupby('region')['article_cleaned'].apply(lambda x: np.mean(x.apply(lambd
print("\nNumber of words per sample for each region:")
print(result)
```

```
Number of words per sample for each region:
region
central       426.266667
east-coast    836.000000
west-coast    623.733333
Name: article_cleaned, dtype: float64
```

In [85]:
```
sns.set(style="whitegrid")
# Create a KDE plot with different visualizations based on region
plt.figure(figsize=(12, 6))
sns.kdeplot(data=df, x=[len(s) for s in df['article_cleaned']], hue=df['region'], fill
```

Out[85]:
```
<Axes: ylabel='Density'>
```



Based on the distribution plot above, we can infer that the articles in `central`` were mostly in the range of same length while the articles from east coast were of varied lengths.

In [86]:
```
# finding the frequently used words in articles for each year
from collections import Counter

my_dict = {}

for year in df['year'].unique():
    text_year = ' '.join(df[df['year'] == year]['article_cleaned'])
    all_words = text_year.lower().split()

    word_counts = Counter(all_words)
    sorted_items = dict(sorted(word_counts.items(), key=lambda item: item[1], reverse=

    top_keys = list(sorted_items.keys())[:10]
    my_dict[year] = top_keys

print("The top 10 words frequently used in the articles published each year include th
my_dict
```

```
The top 10 words frequently used in the articles published each year include the follo
wing:
```

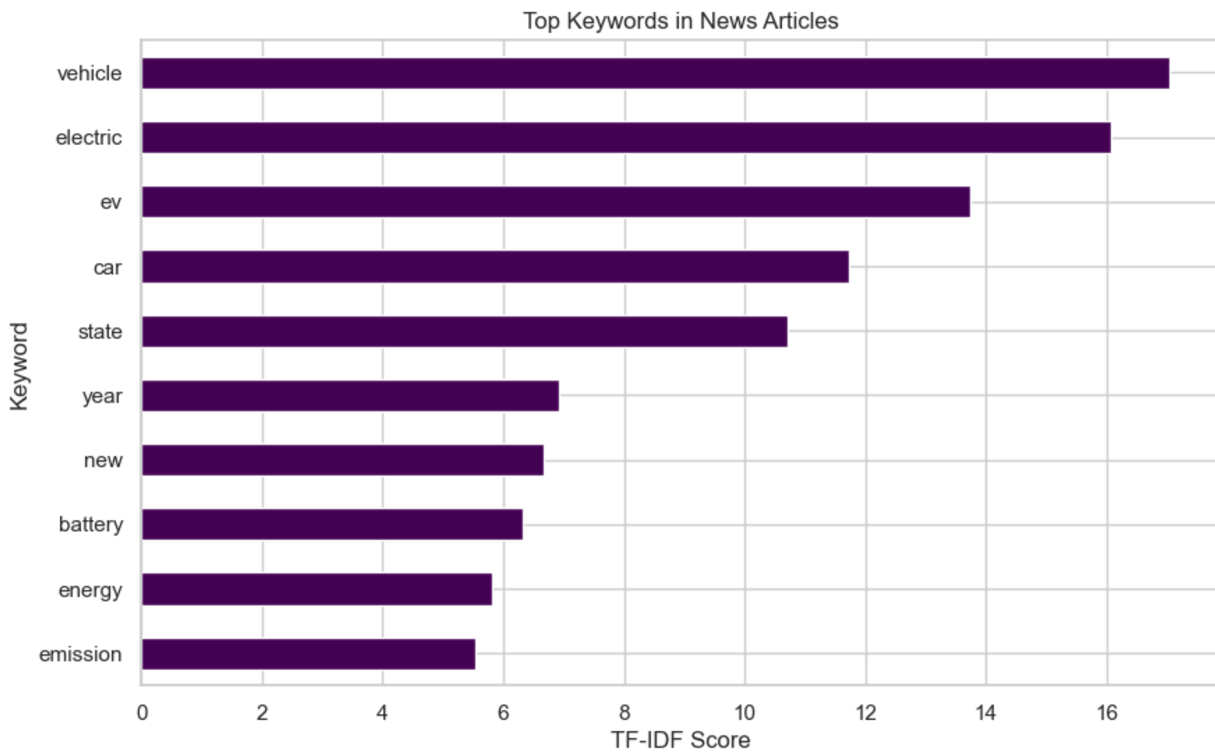
```
Out[86]: {2022: ['electric',
  'car',
  'ev',
  'vehicle',
  'credit',
  'battery',
  'not',
  'would',
  'state',
  'year'],
 2023: ['vehicle',
  'electric',
  'ev',
  'car',
  'new',
  'energy',
  'year',
  'battery',
  'charging',
  'would'],
 2021: ['electric',
  'vehicle',
  'car',
  'emission',
  'cost',
  'climate',
  'state',
  'price',
  'would',
  'new'],
 2019: ['vehicle',
  'city',
  'electric',
  'car',
  'san',
  'tax',
  'francisco',
  'charging',
  'state',
  'would'],
 2020: ['charging',
  'transportation',
  'install',
  'station',
  'county',
  'project',
  'level',
  '2',
  'charger',
  'public'],
 2015: ['tax',
  'electric',
  'vehicle',
  'exemption',
  'sale',
  'would',
  'state',
  'year',
  'carlyle',
  'car'],
```

```
2018: ['ev',
 'fleet',
 'state',
 'vehicle',
 'law',
 'government',
 'washington',
 'public',
 'local',
 'county'],
2017: ['electric',
 'city',
 'vehicle',
 'emission',
 'car',
 'houston',
 'leaf',
 'charging',
 'station',
 'state']}
```

Keyword Extraction

In [87]:

```
"""
This code uses TF-IDF to extract and visualize the top keywords from the news articles
"""

from sklearn.feature_extraction.text import TfidfVectorizer
corpus = df['article_cleaned']

# Create a TF-IDF vectorizer
vectorizer = TfidfVectorizer(max_features=10, stop_words='english')

# Fit and transform the corpus
tfidf_matrix = vectorizer.fit_transform(corpus)

# Get feature names (keywords)
feature_names = vectorizer.get_feature_names_out()

# Create a DataFrame to display the top keywords
keywords_df = pd.DataFrame(tfidf_matrix.toarray(), columns=feature_names)

# Plot the top keywords
plt.figure(figsize=(10, 6))
keywords_df.sum().sort_values().plot(kind='barh', colormap='viridis')
plt.title('Top Keywords in News Articles')
plt.xlabel('TF-IDF Score')
plt.ylabel('Keyword')
plt.show()
```



Indepth analysis of above Visualization

The keywords are ranked by their TF-IDF scores, which measure the importance of a keyword to a document relative to the other documents in the corpus.

The top 3 keywords are:

vehicle

electric

ev

hese keywords suggest that news articles about EVs are focusing on the following topics:

The development of new EV models and the adoption of EVs by consumers. The environmental benefits of EVs, such as their reduced emissions. Government policies that support the adoption of EVs.

The presence of the keyword "emission" in the list of top keywords suggests that consumers are increasingly aware of the environmental benefits of EVs.

The presence of the keyword "state" in the list of top keywords suggests that government policies are playing a role in the adoption of EVs.

In [88]:

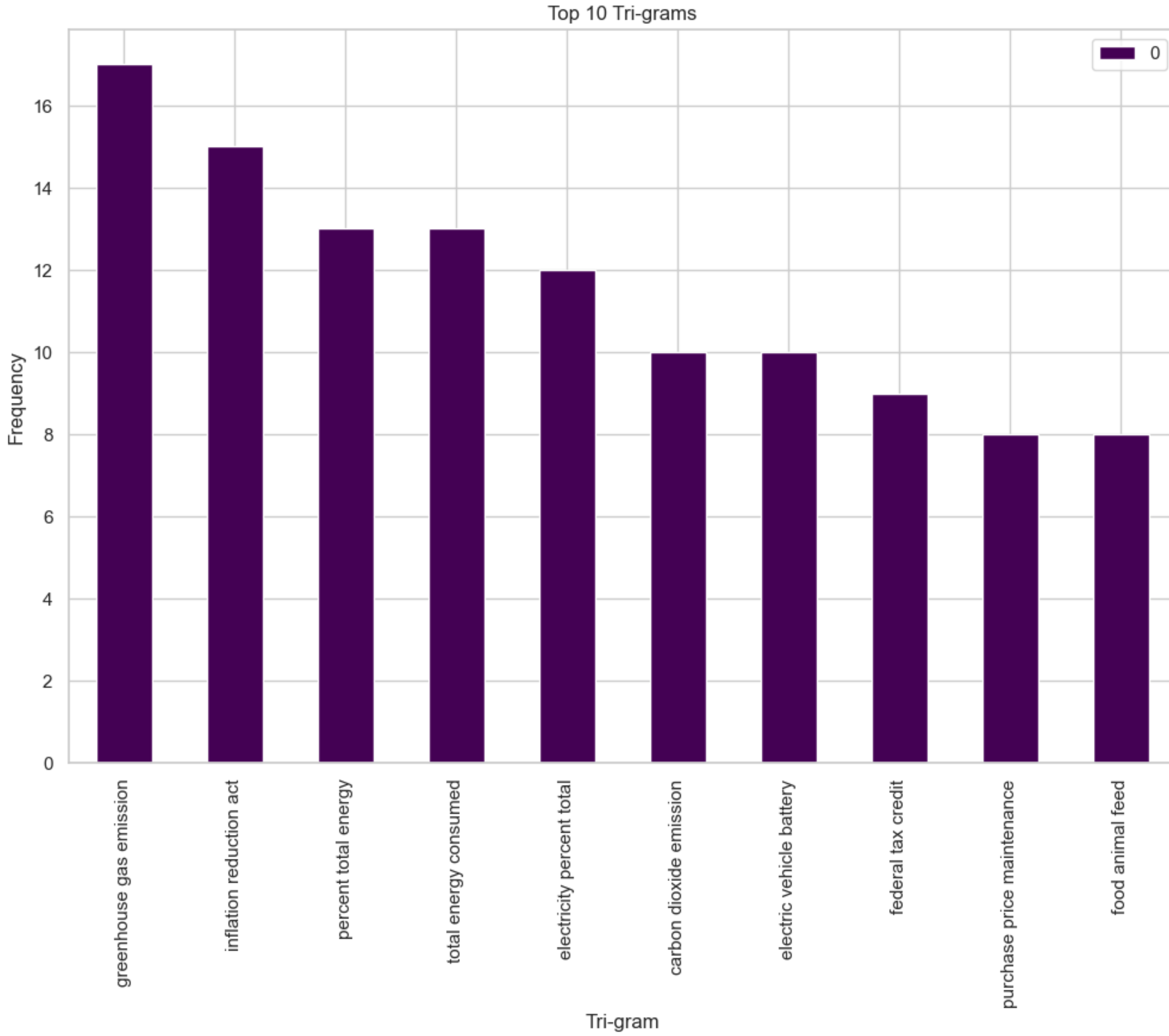
```
#Top 5 Words in News Articles by Year with Count

# Create an empty DataFrame to store the results
ndf = pd.DataFrame(columns=['Year', 'Top_Word', 'Count'])
```

```
for year in df['year'].unique():
    text_year = ' '.join(df[df['year'] == year]['article_cleaned'])
    all_words = text_year.split()
    Freq_word = {}
    for w in all_words:
        w1 = w.lower()
        if w1 in Freq_word:
            Freq_word[w1] += 1
        else:
            Freq_word[w1] = 1
    sorted_items = dict(sorted(Freq_word.items(), key=lambda item: item[1], reverse=Tr
    top_keys = list(sorted_items.keys())[:5]

    # Append results to the new DataFrame
    year_df = pd.DataFrame({'Year': [year] * 5, 'Top_Word': top_keys, 'Count': list(so
    ndf = pd.concat([ndf, year_df], ignore_index=True)

# Sort the DataFrame to control the order in the visualization
ndf = ndf.sort_values(by=['Year', 'Count'], ascending=[True, False])

# bar plot using plotly express
fig = px.bar(ndf, x='Top_Word', y='Count', color='Year',
             title='Top 5 Words in News Articles by Year with Count',
             labels={'Top_Word': 'Top Word', 'Count': 'Count', 'Year': 'Year'},
             height=600)

# Adjust layout for better readability
fig.update_layout(
    xaxis_title='Top Words',
    yaxis_title='Count',
    legend_title_text='Year',
    margin=dict(l=20, r=20, t=40, b=40),
)
fig.show()
```

Based on the visualisation above, we can infer that the electric cars have progressed in a systematic way any product is supposed to progress. Satrting with financial considerations in 2015 to state and law consideration in 2018. From charging station considerations in 2020 to just the discussion about electric vehicles and new possibilities in 2022-23.

In [89]:
```
# code to analyze the frequency of three consecutive words (trigrams) in the text corp
vectorizer = CountVectorizer(ngram_range=(3, 3))
ngrams = vectorizer.fit_transform(df['article_cleaned'])
```

In [91]:
```
ngrams_freq = pd.DataFrame(ngrams.sum(axis=0), columns=vectorizer.get_feature_names_ou
ngrams_freq.head(10).plot(kind='bar', figsize=(12, 8), colormap='viridis')
plt.title('Top 10 Tri-grams')
plt.xlabel('Tri-gram')
plt.ylabel('Frequency')
plt.show()
```



Indepth analysis of above Visualization

A tri-gram is a sequence of three words, such as "greenhouse gas emission" or "electric vehicle battery." The tri-grams in the graph are ranked by their frequency, which is the number of times they appear in the dataset. Based on the data above, mostly the article focussed on climate change considerations and energy consumption statistics, which is the most reasonable way to respresent electric vehicles. The highest written topic being greenhouse gas emissions, which shows the media was mostly trying to foster the production of electrical vehicles.

These tri-grams suggest that the following topics are being covered in news articles about EVs:

The environmental benefits of EVs, such as their reduced greenhouse gas emissions. The cost of EVs, including the price of the batteries and the federal tax credit. The impact of EVs on the energy sector, such as the percentage of total energy consumed by EVs. The performance of EVs, such as their range and the energy efficiency of their batteries. The adoption of EVs in different sectors of the economy, such as the light vehicle market.

The presence of the tri-gram "food animal feed" in the list of top 10 tri-grams suggests that news articles about EVs are also covering the impact of EVs on agriculture. This is likely due to the fact that EVs are becoming more popular in the agricultural sector, as they can help farmers reduce their fuel costs and emissions.

The presence of the tri-gram "electricity percent total" in the list of top 10 tri-grams suggests that there is a growing interest in the impact of EVs on the electricity grid.

The presence of the tri-gram "federal tax credit" in the list of top 10 tri-grams suggests that government policies are playing a role in the adoption of EVs. The presence of the tri-gram "light vehicle market" in the list of top 10 tri-grams suggests that EVs are becoming more popular in the passenger vehicle market.

In [92]:
```
# Load spaCy model for NER
nlp = spacy.load("en_core_web_sm")
```

In [93]:
```
# Create a new column 'entities' to store the extracted entities
df['entities'] = df['article_cleaned'].apply(lambda text: [(ent.text, ent.label_) for
```

This column is to store information about named entities detected in the 'article_cleaned' column of the DataFrame. it contains identified entities and their corresponding labels (e.g., person, organization, location).

In [94]:
```
df.head()
```

Out[94]:

	title	article	news_source	region	article_cleaned	converted_date	y
article_no							
0	Commentary: Driving an EV does not make you p...	['When I started driving an electric vehicle i...	latimes	west-coast	started driving electric vehicle 2018 became p...	09-17-2022	2(
1	Op-Ed: Think bigger. Switching to electric ca...	['It might feel like the easy solution — just ...	latimes	west-coast	might feel like easy solution replace gasguzzl...	09-15-2022	2(
2	Editorial: EPA wants to speed up EV switch. G...	['The Biden administration just proposed hitti...	latimes	west-coast	biden administration proposed hitting accelera...	04-12-2023	2(
3	California's electric car revolution, designe...	['The precious cargo on the ship docked in San...	latimes	west-coast	precious cargo ship docked san diego bay strik...	07-21-2021	2(
4	Electric cars now make up a fifth of Californ...	['One out of every 5 cars sold in California i...	latimes	west-coast	one every 5 car sold california powered batter...	11-01-2023	2(

In [95]:
```python
df['entities'][0]
```

Out[95]:
```
[('2018', 'DATE'),
 ('100', 'CARDINAL'),
 ('2035', 'DATE'),
 ('2018', 'DATE'),
 ('la el', 'GPE'),
 ('five hour day', 'TIME'),
 ('nissan', 'ORG'),
 ('60mile', 'CARDINAL'),
 ('daily', 'DATE'),
 ('200', 'CARDINAL'),
 ('los angeles', 'GPE'),
 ('294', 'CARDINAL'),
 ('2021', 'CARDINAL'),
 ('american', 'NORP'),
 ('one', 'CARDINAL'),
 ('one', 'CARDINAL'),
 ('los angeles', 'GPE'),
 ('nearly 10000', 'CARDINAL'),
 ('california', 'GPE')]
```

In [96]:
```python
# this code extracts information about named entities from the DataFrame (df) and orga

# Create an empty list to store DataFrames
dfs = []

# Iterate through each row in the original DataFrame
for idx, row in df.iterrows():
    year = row['year']
    entities = row['entities']

    # Create a DataFrame for the current row's entities
    entity_df_row = pd.DataFrame({'year': [year]*len(entities),
                                  'entity': [entity[0] for entity in entities],
                                  'label': [entity[1] for entity in entities]})

    # Append the DataFrame to the list
    dfs.append(entity_df_row)

# Concatenate all DataFrames in the list into a single DataFrame
entity_df = pd.concat(dfs, ignore_index=True)

# Reset index for clarity
entity_df.reset_index(drop=True, inplace=True)
```

In [97]:
```python
entity_df.head(10)
```

Out[97]:

	year	entity	label
0	2022	2018	DATE
1	2022	100	CARDINAL
2	2022	2035	DATE
3	2022	2018	DATE
4	2022	la el	GPE
5	2022	five hour day	TIME
6	2022	nissan	ORG
7	2022	60mile	CARDINAL
8	2022	daily	DATE
9	2022	200	CARDINAL

In [98]:

```
entity_df['year'].unique()
```

Out[98]:

```
array([2022, 2023, 2021, 2019, 2020, 2015, 2018, 2017], dtype=int64)
```

In [99]:

```
entity_df['label'].unique()
```

Out[99]:

```
array(['DATE', 'CARDINAL', 'GPE', 'TIME', 'ORG', 'NORP', 'QUANTITY',
       'LOC', 'ORDINAL', 'PERSON', 'MONEY', 'PRODUCT', 'PERCENT', 'FAC',
       'WORK_OF_ART', 'EVENT', 'LANGUAGE'], dtype=object)
```

In [100…

```
# Extract 'year' and 'ORG' columns
org_df = entity_df[['year', 'entity','label']]

# Filter rows with 'ORG' label
org_df = org_df[org_df['label'] == 'ORG']

# Flatten the list of ORG entities
org_df = org_df.explode('entity')
```

In [101…

```
org_df.head()
```

Out[101]:

	year	entity	label
6	2022	nissan	ORG
45	2023	biden administration	ORG
48	2023	epa	ORG
53	2023	van	ORG
55	2023	epa	ORG

In [102…

```
# Count occurrences of each ORG in each year
org_counts_by_year = org_df.groupby(['year', 'entity']).size().reset_index(name='Count

# Display the top 5 most occurred ORG in each year
top_orgs_by_year = org_counts_by_year.groupby('year').apply(lambda x: x.nlargest(5, 'C
```

```
top_orgs_by_year.rename(columns={'entity': 'ORG'}, inplace=True)
top_orgs_by_year.head()
```

Out[102]:

	year	ORG	Count
0	2015	nissan	3
1	2015	hill	1
2	2015	mann	1
3	2015	ritzville house	1
4	2015	senate	1

In [103…

```
#Top 5 Most Occurred ORG in Each Year
fig = px.bar(top_orgs_by_year, x='Count', y='ORG', color='year',
             labels={'Count': 'Occurrences'},
             title='Top 5 Most Occurred ORG in Each Year',
             category_orders={"year": sorted(top_orgs_by_year['year'], reverse=True)})
fig.update_layout(
    width=1000,
    height=700,
)
fig.show()
```

From the above visualization we can infer that, along the years the organizations ford and nissan were under discussion on social media throughout.During the later years companies like hyundai and kia came into discussion. In the initial years mostly some resources, batteries and energy industries were reported in media.

Nissan has been the most frequently mentioned organization in recent years. This suggests that Nissan is a major player in the electric vehicle industry.

There is a growing interest in electric vehicles in the United States. The fact that the number of mentions of electric vehicle-related organizations has increased steadily over the past few years suggests that there is a growing interest in electric vehicles in the United States.

The presence of Pasadena Houston in the list of top 5 most occurred organizations in 2017 suggests that there is a growing interest in electric vehicles in the southern United States.

The presence of Senate in the list of top 5 most occurred organizations in 2015 and 2022 suggests that the government is considering policies related to electric vehicles.

In [104…

```
entity_df[entity_df['label']=='PERSON']['entity'].value_counts().head(10)
```

Out[104]:

```
entity
joe bidens       7
chris            5
mustang mache    5
mustang mach     3
brian kemp       2
joe biden        2
anderson         2
kelley           2
davis            2
melvin carter    2
Name: count, dtype: int64
```

The above data represents the people that were most mentioned in the articles during the years. This can inform how the electric vehicles progressed over years as the president was mentioned in the articles the most times. This means the electric vehicles came under the discussion of country's topics very often.

In [107…

```
# Count entity occurrences
entity_counts = {}
for entities_list in df['entities']:
    for entity, _ in entities_list:
        entity_counts[entity] = entity_counts.get(entity, 0) + 1

# Choose top 15 entities based on frequency for whole articles
top_entities = [entity for entity, count in sorted(entity_counts.items(), key=lambda x
```

In [113…

```
top_entities
```

Out[113]:

```
['one',
 'california',
 'ford',
 'united states',
 'nissan',
 'today',
 '2021',
 'maryland',
 'first',
 'north america',
 'washington',
 '7500',
 'illinois',
 'year',
 'texas']
```

Cross-Region Comparison

In [108…

```
"""
 Creates a stacked bar chart where each bar represents a region,
 and the segments within the bar represent the counts of different entities.
 It provides a visual comparison of the distribution of top entities across regions.
"""
# distribution of the top entities in each region

# Create a DataFrame to store entity counts for each region
region_entity_counts = pd.DataFrame(index=top_entities, columns=df['region'].unique())

# Fill the DataFrame with entity counts
for region in df['region'].unique():
    region_df = df[df['region'] == region]
    entity_counts = {}
    for entities_list in region_df['entities']:
        for entity, _ in entities_list:
            entity_counts[entity] = entity_counts.get(entity, 0) + 1
    region_entity_counts[region] = region_entity_counts.index.map(entity_counts)

# Transpose the DataFrame for better visualization
region_entity_counts = region_entity_counts.T

# Convert the DataFrame to long format for Plotly
region_entity_counts_long = region_entity_counts.reset_index().melt(id_vars='index', v

# Create an interactive stacked bar chart with hover information
fig = px.bar(region_entity_counts_long, x='index', y='Entity Count', color='Entity', b
             labels={'index': 'Region', 'Entity Count': 'Entity Count'},
             title='Top Entities Distribution Across Regions',
             hover_data={'Entity': True, 'Entity Count': True})

# Show the interactive plot
fig.show()
```

In [116…

```
region_entity_counts_long
```

Out[116]:

	index	Entity	Entity Count
0	west-coast	one	21.0
1	east-coast	one	25.0
2	central	one	13.0
3	west-coast	california	40.0
4	east-coast	california	14.0
5	central	california	2.0
6	west-coast	ford	3.0
7	east-coast	ford	26.0
8	central	ford	5.0
9	west-coast	united states	3.0
10	east-coast	united states	24.0
11	central	united states	1.0
12	west-coast	nissan	5.0
13	east-coast	nissan	14.0
14	central	nissan	7.0
15	west-coast	today	6.0
16	east-coast	today	15.0
17	central	today	4.0
18	west-coast	2021	2.0
19	east-coast	2021	17.0
20	central	2021	5.0
21	west-coast	maryland	NaN
22	east-coast	maryland	23.0
23	central	maryland	NaN
24	west-coast	first	7.0
25	east-coast	first	12.0
26	central	first	3.0
27	west-coast	north america	NaN
28	east-coast	north america	11.0
29	central	north america	11.0
30	west-coast	washington	16.0
31	east-coast	washington	3.0
32	central	washington	1.0
33	west-coast	7500	NaN

	index	Entity	Entity Count
34	east-coast	7500	10.0
35	central	7500	10.0
36	west-coast	illinois	NaN
37	east-coast	illinois	1.0
38	central	illinois	18.0
39	west-coast	year	4.0
40	east-coast	year	10.0
41	central	year	3.0
42	west-coast	texas	NaN
43	east-coast	texas	1.0
44	central	texas	16.0

From the above vizualization: California is more frequently mentioned on the West Coast, which is expected given its location. Ford is prominently mentioned on the East Coast. The mentions of "North America" are more concentrated on the East Coast.

In [54]:
```
print(region_entity_counts_long)
```

```
         index         Entity  Entity Count
0   west-coast     california          42.0
1   east-coast     california          14.0
2      central     california           1.0
3   west-coast            one          18.0
4   east-coast            one          18.0
5      central            one          10.0
6   west-coast           ford           3.0
7   east-coast           ford          26.0
8      central           ford           5.0
9   west-coast  united states           4.0
10  east-coast  united states          24.0
11     central  united states           1.0
12  west-coast         nissan           5.0
13  east-coast         nissan          14.0
14     central         nissan           7.0
15  west-coast           2021           3.0
16  east-coast           2021          17.0
17     central           2021           5.0
18  west-coast          first           6.0
19  east-coast          first          12.0
20     central          first           4.0
21  west-coast  north america           NaN
22  east-coast  north america          11.0
23     central  north america          11.0
24  west-coast           7500           NaN
25  east-coast           7500          10.0
26     central           7500          10.0
27  west-coast       maryland           NaN
28  east-coast       maryland          20.0
29     central       maryland           NaN
30  west-coast          today           6.0
31  east-coast          today          10.0
32     central          today           3.0
33  west-coast     washington          15.0
34  east-coast     washington           3.0
35     central     washington           1.0
36  west-coast         toyota           2.0
37  east-coast         toyota          15.0
38     central         toyota           2.0
39  west-coast           2030          11.0
40  east-coast           2030           2.0
41     central           2030           5.0
42  west-coast       illinois           NaN
43  east-coast       illinois           1.0
44     central       illinois          17.0
```

Topic Modeling

In [117…

```
#This code performs topic modeling on text data using Latent Dirichlet Allocation (LDA
# 'article_cleaned' columns already contains the text that is lemmatized and cleaned.
region_topics = {}

for region, region_df in df.groupby('region'):
    print(region)
    # 'article_cleaned' column already contains preprocessed text
    tokens = [word_tokenize(text) for text in region_df['article_cleaned']]

    # Create a dictionary and a corpus for each region
```

```
    # mapping between words and their integer IDs(each unique word is assigned a uniqu
    dictionary = Dictionary(tokens)

    # each document is represented as a list of word IDs along with their frequencies
    corpus = [dictionary.doc2bow(token_list) for token_list in tokens]

    # LDA Model set to iterate 15 times over the entire corpus
    lda_model = LdaModel(corpus, num_topics=5, id2word=dictionary, passes=15, random_s

    # Get topics
    topics = lda_model.show_topics(num_words=10, formatted=False)

    region_topics[region] = topics
```

```
central
east-coast
west-coast
```

In [118…

```
print(type(tokens))
```

```
<class 'list'>
```

In [119…

```
print(tokens[0])
```

```
['started', 'driving', 'electric', 'vehicle', '2018', 'became', 'part', 'problem', 're
ason', 'cited', 'ev', 'critic', 'recent', 'heat', 'wave', 'state', 'asked', 'electri
c', 'car', 'charged', 'peak', 'demand', 'prompted', 'howl', 'told', 'think', 'electrif
ication', 'everything', 'home', 'appliance', 'car', 'leftwing', 'pipe', 'dream', 'espe
cially', 'light', 'californias', 'mandate', 'requiring', '100', 'new', 'vehicle', 'sal
e', 'zeroemission', '2035', 'part', 'problem', 'worry', 'expressed', 'ad', 'nauseum',
'ev', 'skeptic', 'much', 'merit', 'yes', 'driving', 'distance', 'issue', 'tiny', 'perc
entage', 'trip', 'yes', 'electricity', 'bill', 'higher', 'additional', 'cost', 'far',
'lower', 'driver', 'pay', 'gas', 'yes', 'apartment', 'condo', 'dweller', 'plug', 'nigh
t', 'legitimate', 'concern', 'must', 'rely', 'public', 'charging', 'infrastructure',
'growing', 'yes', 'ev', 'carbon', 'footprint', 'still', 'typically', 'much', 'smalle
r', 'gas', 'car', 'evidence', 'compelling', 'hard', 'imagine', 'survival', 'planet',
'unless', 'immediately', 'replace', 'gaspowered', 'auto', 'electric', 'one', 'conseque
ntly', 'state', 'federal', 'government', 'want', 'take', 'meaningful', 'action', 'clim
ate', 'change', 'choice', 'subsidize', 'ev', 'buyer', 'that', 'is', 'part', 'problem',
'end', 'electric', 'car', 'still', 'well', 'car', 'mass', 'car', 'ownership', 'devasta
ting', 'environmental', 'consequence', 'beyond', 'tailpipe', 'emission', 'became', 'pa
rt', 'car', 'culture', '2018', 'times', 'moved', 'downtown', 'la', 'el', 'segundo', 'd
edicated', 'transit', 'commuter', 'even', 'held', 'month', 'times', 'relocation', 'fiv
e', 'hour', 'day', 'bus', 'train', 'eventually', 'got', 'leased', 'nissan', 'leaf', 'f
unny', 'thing', 'happened', 'wellmeaning', 'people', 'reacted', 'done', 'world', 'favo
r', 'never', 'mind', 'loss', 'transit', 'user', '60mile', 'daily', 'drive', 'become',
'ev', 'operator', 'pitied', 'bus', 'riding', 'praised', 'driving', 'electric', 'vehicl
e', 'like', 'gaspowered', 'car', 'require', 'vast', 'expanse', 'concrete', 'asphalt',
'automotive', 'use', 'paving', 'entire', 'region', 'turned', 'neighborhood', 'heat',
'sink', 'soak', 'energy', 'sun', 'day', 'release', 'night', 'exactly', 'want', 'era',
'accelerating', 'climate', 'change', 'electric', 'vehicle', 'like', 'gaspowered', 'ca
r', 'force', 'driver', 'sit', 'traffic', 'jam', 'everyone', 'else', 'often', 'freewa
y', 'required', 'bulldozing', 'longestablished', 'minority', 'community', 'built', 'do
wney', 'local', 'fighting', 'highway', 'expansion', 'plan', 'would', 'displace', 'resi
dent', '200', 'home', 'have', 'not', 'asked', 'something', 'tell', 'yeah', 'electric',
'car', 'would', 'not', 'convince', 'resident', 'give', 'fighting', 'home', 'electric',
'vehicle', 'like', 'gaspowered', 'car', 'needlessly', 'kill', 'people', 'city', 'los',
'angeles', 'record', '294', 'people', 'killed', 'traffic', '2021', 'inadequate', 'infr
astructure', 'largely', 'blame', 'increasing', 'size', 'car', 'electric', 'car', 'gett
ing', 'bigger', 'make', 'appealing', 'american', 'consumer', 'is', 'not', 'say', 'ar
e', 'not', 'benefit', 'replacing', 'dirty', 'vehicle', 'zeroemission', 'one', 'citie
s', 'need', 'curb', 'air', 'pollution', 'electric', 'car', 'because', 'little', 'child
ren', 'living', 'near', 'freeway', 'higher', 'rate', 'asthma', 'trading', 'internal',
'combustion', 'engine', 'electric', 'motor', 'would', 'certainly', 'help', 'replacin
g', 'one', 'kind', 'car', 'another', 'is', 'not', 'enough', 'city', 'like', 'los', 'an
geles', 'want', 'anything', 'trafficchoked', 'dystopia', 'right', 'subsidizing', 'ev',
'ownership', 'tune', 'nearly', '10000', 'car', 'sticker', 'price', 'california', 'ma
y', 'necessary', 'reduce', 'acceleration', 'climate', 'change', 'also', 'bandaid', 'bu
y', 'time', 'systemic', 'change', 'take', 'hold', 'kind', 'systemic', 'change', 'woul
d', 'build', 'big', 'public', 'transit', 'system', 'la', 'trying', 'make', 'free', 're
liable', 'safe', 'subsidize', 'purchase', 'electric', 'bike', 'make', 'easier', 'commu
te', 'longer', 'distance', 'device', 'use', 'considerably', 'le', 'power', 'road', 'sp
ace', 'electric', 'car', 'think', 'people', 'neighborhood', 'need', 'people', 'drivin
g', 'neighborhood', 'want', 'love', 'god', 'shelter', 'every', 'bus', 'stop', 'every',
'block', 'rider', 'do', 'not', 'risk', 'sunstroke', 'take', 'transit', 'heat', 'wave']
```

In [120…
```
len(tokens[0])
```

Out[120]:
```
441
```

In [121…
```
len(df['article_cleaned'][0].split())
```

Out[121]:
```
441
```

In [122…

```
print(type(corpus))
```

```
<class 'list'>
```

In [123…

```
# each tuple contains a word ID and its frequency in the document (document-term matri
print(corpus[0])
```

```
[(0, 1), (1, 1), (2, 1), (3, 2), (4, 1), (5, 1), (6, 1), (7, 1), (8, 1), (9, 1), (10,
1), (11, 1), (12, 1), (13, 1), (14, 1), (15, 1), (16, 2), (17, 1), (18, 1), (19, 1),
(20, 1), (21, 1), (22, 1), (23, 2), (24, 1), (25, 1), (26, 1), (27, 1), (28, 1), (29,
2), (30, 1), (31, 1), (32, 1), (33, 1), (34, 1), (35, 1), (36, 1), (37, 1), (38, 1),
(39, 1), (40, 1), (41, 1), (42, 1), (43, 3), (44, 1), (45, 1), (46, 1), (47, 1), (48,
17), (49, 1), (50, 1), (51, 5), (52, 1), (53, 1), (54, 1), (55, 1), (56, 1), (57, 1),
(58, 2), (59, 3), (60, 1), (61, 1), (62, 1), (63, 1), (64, 1), (65, 1), (66, 1), (67,
1), (68, 1), (69, 1), (70, 1), (71, 1), (72, 1), (73, 1), (74, 1), (75, 1), (76, 1),
(77, 1), (78, 2), (79, 1), (80, 1), (81, 1), (82, 1), (83, 1), (84, 1), (85, 2), (86,
1), (87, 1), (88, 1), (89, 1), (90, 1), (91, 1), (92, 2), (93, 4), (94, 1), (95, 1),
(96, 1), (97, 1), (98, 13), (99, 1), (100, 1), (101, 1), (102, 1), (103, 1), (104, 1),
(105, 1), (106, 1), (107, 1), (108, 1), (109, 1), (110, 1), (111, 6), (112, 1), (113,
1), (114, 2), (115, 1), (116, 1), (117, 1), (118, 1), (119, 1), (120, 1), (121, 1), (1
22, 1), (123, 1), (124, 1), (125, 2), (126, 1), (127, 1), (128, 1), (129, 1), (130,
2), (131, 1), (132, 2), (133, 4), (134, 1), (135, 1), (136, 1), (137, 1), (138, 1), (1
39, 1), (140, 1), (141, 1), (142, 1), (143, 3), (144, 1), (145, 1), (146, 2), (147,
1), (148, 1), (149, 3), (150, 1), (151, 1), (152, 1), (153, 1), (154, 1), (155, 1), (1
56, 2), (157, 1), (158, 3), (159, 1), (160, 1), (161, 1), (162, 1), (163, 2), (164,
2), (165, 1), (166, 1), (167, 1), (168, 1), (169, 1), (170, 1), (171, 1), (172, 4), (1
73, 1), (174, 1), (175, 1), (176, 1), (177, 1), (178, 2), (179, 1), (180, 1), (181,
1), (182, 3), (183, 1), (184, 1), (185, 1), (186, 1), (187, 1), (188, 1), (189, 1), (1
90, 1), (191, 1), (192, 1), (193, 2), (194, 1), (195, 1), (196, 1), (197, 1), (198,
1), (199, 2), (200, 1), (201, 3), (202, 1), (203, 1), (204, 2), (205, 1), (206, 6), (2
07, 1), (208, 3), (209, 1), (210, 2), (211, 4), (212, 1), (213, 1), (214, 1), (215,
5), (216, 1), (217, 1), (218, 1), (219, 1), (220, 1), (221, 1), (222, 1), (223, 1), (2
24, 1), (225, 1), (226, 3), (227, 1), (228, 2), (229, 1), (230, 1), (231, 1), (232,
1), (233, 1), (234, 1), (235, 1), (236, 1), (237, 1), (238, 1), (239, 1), (240, 1), (2
41, 1), (242, 2), (243, 1), (244, 1), (245, 1), (246, 2), (247, 1), (248, 1), (249,
1), (250, 1), (251, 1), (252, 1), (253, 1), (254, 1), (255, 1), (256, 1), (257, 1), (2
58, 1), (259, 1), (260, 1), (261, 1), (262, 1), (263, 1), (264, 1), (265, 1), (266,
2), (267, 1), (268, 2), (269, 1), (270, 2), (271, 1), (272, 1), (273, 1), (274, 1), (2
75, 1), (276, 2), (277, 1), (278, 3), (279, 1), (280, 1), (281, 1), (282, 2), (283,
1), (284, 2), (285, 1), (286, 1), (287, 1), (288, 2), (289, 1), (290, 1), (291, 4), (2
92, 1), (293, 1), (294, 1), (295, 1), (296, 1), (297, 1), (298, 2), (299, 1), (300,
1), (301, 6), (302, 4), (303, 2), (304, 1), (305, 1), (306, 1), (307, 1), (308, 4), (3
09, 1), (310, 4), (311, 2)]
```

In [124…

```
# validation
test_id = 3
token_name = dictionary.get(test_id)
print(f"The token name for ID {test_id} is: {token_name}")
```

```
The token name for ID 3 is: 2018
```

In [125…

```
print(len(corpus[0]))
```

```
312
```

In [126…

```
print(dictionary,type(dictionary))
```

```
Dictionary<3253 unique tokens: ['100', '10000', '200', '2018', '2021']...> <class 'gen
sim.corpora.dictionary.Dictionary'>
```

In [129…

```
#The output of an LDA model:
# For each document: a distribution of topics.
# For each topic: a distribution of words.
region_topics
```

```
Out[129]: {'central': [(0,
   [('credit', 0.022888368),
    ('ev', 0.021746237),
    ('vehicle', 0.01610942),
    ('new', 0.011690473),
    ('yous', 0.010701533),
    ('battery', 0.010273367),
    ('tax', 0.010005996),
    ('price', 0.009945768),
    ('year', 0.009442886),
    ('requirement', 0.007976759)]),
  (1,
   [('electric', 0.029036736),
    ('vehicle', 0.024015693),
    ('illinois', 0.013944505),
    ('state', 0.013319463),
    ('pritzker', 0.008291172),
    ('million', 0.0070348284),
    ('year', 0.0064096344),
    ('act', 0.0057803397),
    ('also', 0.005779971),
    ('money', 0.0057795946)]),
  (2,
   [('electric', 0.01878139),
    ('vehicle', 0.017536903),
    ('station', 0.010357312),
    ('charging', 0.009621767),
    ('car', 0.009599668),
    ('ev', 0.0069710985),
    ('state', 0.0066854865),
    ('city', 0.006314366),
    ('emission', 0.0063040187),
    ('texas', 0.0048479987)]),
  (3,
   [('vehicle', 0.0005119617),
    ('ev', 0.00050316),
    ('electric', 0.0004998698),
    ('state', 0.0004642574),
    ('year', 0.00046196484),
    ('charging', 0.00045908598),
    ('car', 0.00045663887),
    ('station', 0.00045640036),
    ('new', 0.0004522223),
    ('would', 0.00044987758)]),
  (4,
   [('ev', 0.014563589),
    ('vehicle', 0.008299691),
    ('charging', 0.0073273224),
    ('energy', 0.006839465),
    ('electric', 0.00636786),
    ('houston', 0.0063566905),
    ('texas', 0.0058778822),
    ('year', 0.0054016877),
    ('station', 0.00539883),
    ('yous', 0.0049172197)])],
 'east-coast': [(0,
   [('vehicle', 0.01760854),
    ('electric', 0.0139635755),
    ('car', 0.012034099),
    ('ev', 0.01115006),
```

```
   ('battery', 0.010070068),
   ('price', 0.009221432),
   ('new', 0.0075977943),
   ('model', 0.0070772157),
   ('year', 0.006136362),
   ('say', 0.0051629604)]),
 (1,
  [('car', 0.017041072),
   ('electric', 0.01349198),
   ('ev', 0.013463708),
   ('new', 0.00858948),
   ('not', 0.007667651),
   ('dealer', 0.0076621994),
   ('vehicle', 0.007207793),
   ('dealership', 0.0067693717),
   ('home', 0.005458246),
   ('sale', 0.0054401713)]),
 (2,
  [('maryland', 0.014785704),
   ('year', 0.011018584),
   ('vehicle', 0.010239101),
   ('ev', 0.009044819),
   ('credit', 0.008951586),
   ('state', 0.008618678),
   ('would', 0.008259131),
   ('electric', 0.007250309),
   ('administration', 0.0061114416),
   ('not', 0.005803319)]),
 (3,
  [('electric', 0.015919998),
   ('energy', 0.012953645),
   ('car', 0.0099988915),
   ('emission', 0.009572425),
   ('vehicle', 0.009570588),
   ('heating', 0.008504287),
   ('climate', 0.008294006),
   ('cost', 0.008090137),
   ('electricity', 0.007237535),
   ('fuel', 0.0072373054)]),
 (4,
  [('ev', 0.011848351),
   ('new', 0.0073773433),
   ('car', 0.0064865584),
   ('vehicle', 0.0064836736),
   ('carbon', 0.0064603533),
   ('emission', 0.0055746026),
   ('gaspowered', 0.0046657645),
   ('would', 0.0037825098),
   ('tax', 0.003773509),
   ('part', 0.0037690285)])],
'west-coast': [(0,
  [('ev', 0.01231523),
   ('fleet', 0.011092003),
   ('charging', 0.010484485),
   ('state', 0.009878781),
   ('county', 0.008653908),
   ('vehicle', 0.0074501047),
   ('public', 0.0074355253),
   ('transportation', 0.0068268715),
   ('station', 0.0062177214),
```

```
    ('install', 0.006216404)]),
  (1,
   [('electric', 0.01586009),
    ('vehicle', 0.015541304),
    ('car', 0.011740527),
    ('state', 0.007647408),
    ('california', 0.00764097),
    ('would', 0.0073289624),
    ('power', 0.006379496),
    ('tax', 0.0054340353),
    ('ev', 0.005433197),
    ('climate', 0.0054304497)]),
  (2,
   [('vehicle', 0.017551579),
    ('electric', 0.012309433),
    ('car', 0.011633825),
    ('city', 0.009506511),
    ('would', 0.008810616),
    ('tax', 0.008103158),
    ('sale', 0.007058101),
    ('state', 0.0067165033),
    ('charging', 0.005667051),
    ('gas', 0.005315588)]),
  (3,
   [('electric', 0.00038499423),
    ('car', 0.00038431978),
    ('vehicle', 0.00037602056),
    ('not', 0.00034573252),
    ('would', 0.0003446473),
    ('state', 0.00034273518),
    ('ev', 0.00033501006),
    ('climate', 0.00033500334),
    ('people', 0.00033317684),
    ('change', 0.0003329449)]),
  (4,
   [('electric', 0.013312018),
    ('car', 0.013069473),
    ('vehicle', 0.0067960494),
    ('not', 0.0063268766),
    ('battery', 0.00609489),
    ('ev', 0.005862355),
    ('state', 0.005631825),
    ('lithium', 0.0053954013),
    ('california', 0.0047005396),
    ('change', 0.004466884)])]}
```

for each region, Lists containing tuples, where each tuple represents a topic within that region. and for each tuple inside the list in each region:

Tuple Format: (topic_number, word_probability_list) topic_number: An identifier for the topic within that region. word_probability_list: A list of tuples, where each tuple represents a word and its probability within that topic.

word_probability_list format: Tuple Format: ('word', probability) 'word': The actual word associated with the topic. probability: The probability of that word being part of the topic.

Central Region:

Topic 0: This topic seems to be related to electric vehicle incentives, with words like 'credit,' 'EV,' 'vehicle,' and 'new' being prominent.

Topic 1: Focuses on Illinois state and its actions related to electric vehicles, mentioning 'pritzker,' 'state,' and 'act.'

Topic 2: Involves discussions about electric vehicle infrastructure, including 'charging station,' 'charging,' 'car,' and 'emission.'

Topic 3: Generic terms related to electric vehicles and states, but with lower probabilities.

Topic 4: Discusses various aspects such as energy, Houston, Texas, and charging, indicating a diverse set of words.

East Coast Region:

Topic 0: General terms related to electric vehicles, including 'vehicle,' 'electric,' 'car,' and 'EV.'

Topic 1: Discusses dealerships, sales, and transactions, mentioning 'car,' 'electric,' 'dealer,' and 'dealership.'

Topic 2: Focuses on Maryland, mentioning 'Maryland,' 'year,' 'vehicle,' and 'EV.'

Topic 3: Involves discussions about energy, emissions, and climate, with terms like 'electric,' 'energy,' 'emission,' and 'climate.'

Topic 4: Touches upon new cars, carbon emissions, and taxes, mentioning 'EV,' 'car,' 'vehicle,' 'carbon,' and 'emission.'

West Coast Region:

Topic 0: Involves discussions about fleets, charging, and public transportation, mentioning 'EV,' 'fleet,' 'charging,' and 'public.'

Topic 1: Discusses various aspects related to California, mentioning 'electric,' 'vehicle,' 'car,' and 'state.'

Topic 2: Involves discussions about vehicle sales, taxes, and charging, mentioning 'vehicle,' 'electric,' 'car,' 'tax,' and 'charging.'

Topic 3: Generic terms related to electric vehicles, states, and climate, but with lower probabilities.

Topic 4: Touches upon electric cars, batteries, and lithium, mentioning 'electric,' 'car,' 'vehicle,' 'battery,' 'EV,' and 'lithium.'

In [132…

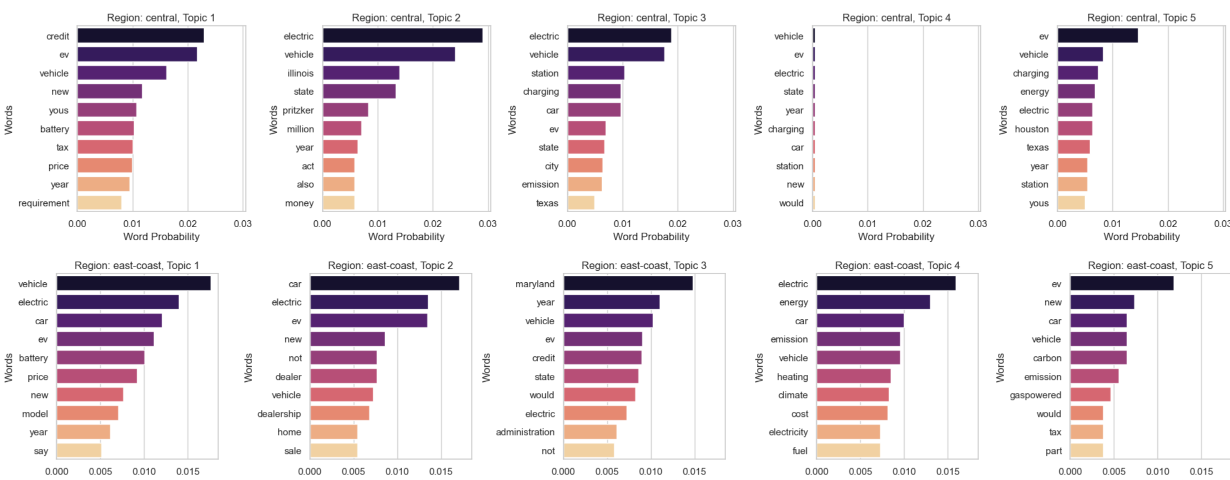
```
# identifying common topics accross regions
from collections import Counter
```

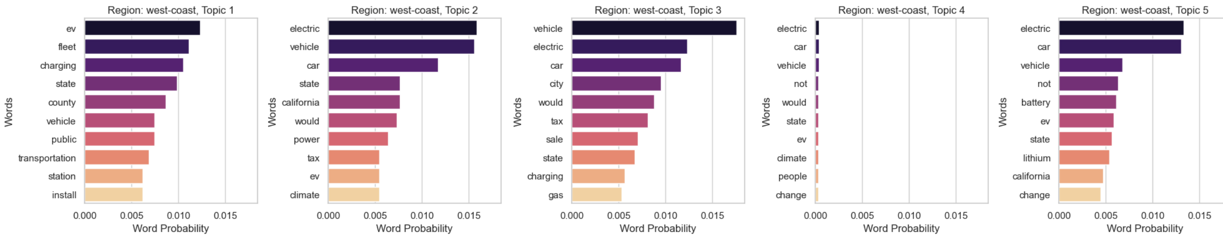
```python
# Extract topics for each region
all_topics = [topic[0] if isinstance(topic[0], str) else topic[1] for region_topics_li

# Flatten the list of topics for each region and remove nested lists
flat_topics = [item for sublist in all_topics if isinstance(sublist, list) for item in

# Count the occurrences of each topic across regions
topic_counts = Counter(flat_topics)

# Extract the top 5 common topics
top_common_topics = [topic for topic, count in topic_counts.most_common(5)]

# Display the top 5 common topics
print("Top 5 Common Topics:")
for topic in top_common_topics:
    print(topic)
```

```
Top 5 Common Topics:
('credit', 0.022888368)
('ev', 0.021746237)
('vehicle', 0.01610942)
('new', 0.011690473)
('yous', 0.010701533)
```

In [137…

```python
# visualization of above topic modelling
for region, topics in region_topics.items():
    fig, axes = plt.subplots(nrows=1, ncols=5, figsize=(20, 4), sharex=True)
    axes = axes.flatten()
    for i, (topic, ax) in enumerate(zip(topics, axes)):
        topic_words = [word for word, _ in topic[1]]
        sns.barplot(x=[count for _, count in topic[1]], y=topic_words, ax=ax, palette=
        ax.set_title(f'Region: {region}, Topic {i + 1}')
        ax.set_xlabel('Word Probability')
        ax.set_ylabel('Words')

    plt.tight_layout()
    plt.show()
```

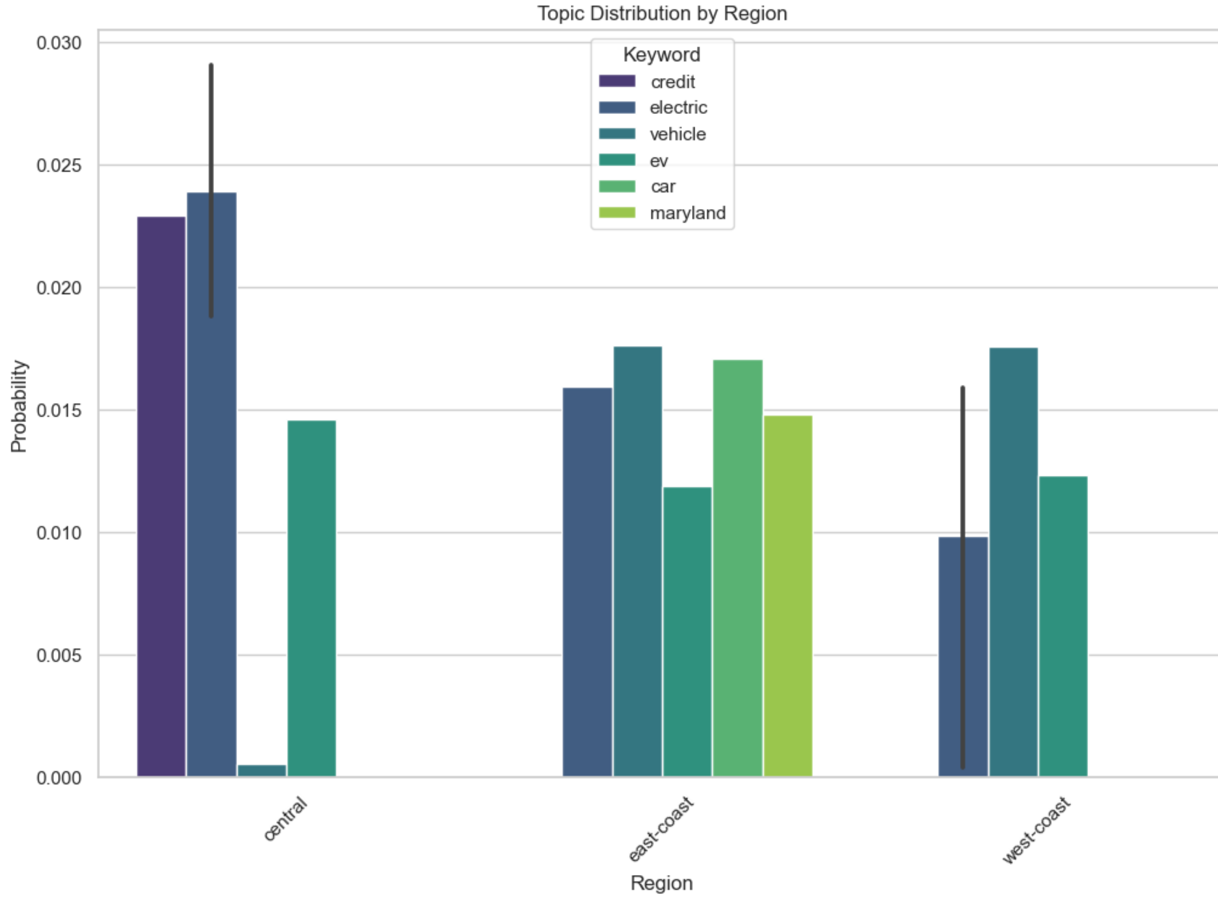




Based on the data above, regarding 5 topics identified in each region. The topics modelled in each region are mostly on the same line. East coast has topics more related to greenouse emissions, west-coast more related to power consumption and Central more related to the places of vehicles and charging stations.

In [138…

```
"""
bar plot showing the distribution of topics by region. Each region is represented by a
and the height of the bars represents the probability of a keyword within a topic.
"""


# Flatten the topics for visualization
flat_topics = [(region, topic, word_prob[0][0], word_prob[0][1]) for region, topics in

# Create a DataFrame for visualization
topics_df = pd.DataFrame(flat_topics, columns=['Region', 'Topic', 'Keyword', 'Probabil

# Plot the distribution of topics by region
plt.figure(figsize=(12, 8))
sns.barplot(x='Region', y='Probability', hue='Keyword', data=topics_df, palette='virid
plt.title('Topic Distribution by Region')
plt.xlabel('Region')
plt.ylabel('Probability')
plt.xticks(rotation=45)
plt.show()
```



In [139…
```
#saving final cleaned ,combined dataframe to csv
df.to_csv('dataset/final_cleaned_data/combined_dataset.csv')
```

In [ ]:

In [ ]: