Week - 12 - Final Submission

Milestone - 1 : Data Selection and EDA
Milestone - 2 : Data Preparation
Milestone - 3 : Model Building and Evaluation
DSC-550
Kalyan Pothineni

Milestone - 1

Introduction

In the evolving landscape of sustainable energy, the accessibility and efficiency of Electric and alternative Fuel Charging Stations play a pivotal role in adopting eco-friendly vehicles. To address this, I am embarking on a data mining project to optimize the placement and accessibility of these stations, ultimately enhancing user experience and promoting the use of electric and alternative fuel vehicles.

As our world transitions towards a greener future, the availability of charging infrastructure is critical. Currently, consumers need help locating nearby charging stations, and station operators often need help efficiently allocating resources. Therefore, this project aims to develop a data-driven model that can strategically place charging stations and provide insights into usage patterns. This will not only improve the overall accessibility of charging stations but also facilitate better urban planning and resource allocation.

Target for this Model

The primary beneficiaries of this model are urban planners, charging station operators, and local government authorities. These stakeholders will benefit from the insights generated by the model in several ways:

1. **Urban Planners**: Urban planners are responsible for designing sustainable and efficient urban environments. They can use the model's recommendations for optimal charging station placement to encourage eco-friendly commuting within cities. By strategically locating stations in areas with high traffic and demand, urban planners can reduce congestion and promote the use of clean energy vehicles.
2. **Charging Station Operators**: Charging station operators face challenges related to station utilization, maintenance, and profitability. By analyzing usage patterns and optimizing station placement, operators can maximize their revenue. Additionally, predictive maintenance insights can help reduce downtime and ensure a seamless charging experience for users.
3. **Local Government Authorities**: Government authorities play a vital role in promoting sustainable transportation. They can leverage the model's recommendations to align with environmental goals and policies. Additionally, insights into station accessibility can help

authorities assess the need for public-private partnerships to improve station access and promote clean energy adoption.

4. **General Public**: Ultimately, the general public stands to benefit the most. With better access to charging infrastructure and optimized station placement, individuals considering electric or alternative fuel vehicles will find it more convenient to make the switch. This not only reduces the carbon footprint but also contributes to a cleaner and healthier environment.

The model aims to bridge the gap between charging station supply and user demand. It will provide actionable insights that empower stakeholders to make data-driven decisions for sustainable urban development and a more accessible and eco-friendly transportation network. By optimizing station placement and promoting accessibility, we hope to accelerate the transition towards a greener and more sustainable future.

Graphical Analysis

In [2]:
```
# Import libraries as needed
import pandas as pd
import matplotlib.pyplot as plt
```

In [3]:
```
# Load the dataset into a dataframe
charging_stations = pd.read_csv('Electric and Alternative Fuel Charging Stations.
                                low_memory=False)
```

In [4]:
```
charging_stations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70406 entries, 0 to 70405
Data columns (total 65 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Fuel Type Code                      70406 non-null  object
 1   Station Name                        70406 non-null  object
 2   Street Address                      70405 non-null  object
 3   Intersection Directions             4493 non-null   object
 4   City                                70406 non-null  object
 5   State                               70406 non-null  object
 6   ZIP                                 70405 non-null  object
 7   Plus4                               0 non-null      float64
 8   Station Phone                       65204 non-null  object
 9   Status Code                         70406 non-null  object
 10  Expected Date                       1218 non-null   object
 11  Groups With Access Code             70406 non-null  object
 12  Access Days Time                    66628 non-null  object
 13  Cards Accepted                      11289 non-null  object
 14  BD Blends                           1217 non-null   object
 15  NG Fill Type Code                   1603 non-null   object
 16  NG PSI                              1597 non-null   object
 17  EV Level1 EVSE Num                 286 non-null    float64
 18  EV Level2 EVSE Num                 54144 non-null  float64
 19  EV DC Fast Count                    8307 non-null   float64
 20  EV Other Info                       49 non-null     object
 21  EV Network                          60907 non-null  object
 22  EV Network Web                      50305 non-null  object
 23  Geocode Status                      70391 non-null  object
 24  Latitude                            70406 non-null  float64
 25  Longitude                           70406 non-null  float64
 26  Date Last Confirmed                 70188 non-null  object
 27  ID                                  70406 non-null  int64
 28  Updated At                          70406 non-null  object
 29  Owner Type Code                     30190 non-null  object
 30  Federal Agency ID                   955 non-null    float64
 31  Federal Agency Name                 955 non-null    object
 32  Open Date                           69310 non-null  object
 33  Hydrogen Status Link                83 non-null     object
 34  NG Vehicle Class                    1780 non-null   object
 35  LPG Primary                         1867 non-null   object
 36  E85 Blender Pump                    4508 non-null   object
 37  EV Connector Types                  60690 non-null  object
 38  Country                             70406 non-null  object
 39  Intersection Directions (French)    188 non-null    object
 40  Access Days Time (French)           6591 non-null   object
 41  BD Blends (French)                  2 non-null      object
 42  Groups With Access Code (French)    70406 non-null  object
 43  Hydrogen Is Retail                  117 non-null    object
 44  Access Code                         70406 non-null  object
 45  Access Detail Code                  7423 non-null   object
 46  Federal Agency Code                 955 non-null    object
 47  Facility Type                       27665 non-null  object
 48  CNG Dispenser Num                   1039 non-null   float64
 49  CNG On-Site Renewable Source        716 non-null    object
 50  CNG Total Compression Capacity      708 non-null    float64
 51  CNG Storage Capacity                357 non-null    float64
```

```
 52  LNG On-Site Renewable Source      62 non-null     object
 53  E85 Other Ethanol Blends          1458 non-null   object
 54  EV Pricing                        16961 non-null  object
 55  EV Pricing (French)               2032 non-null   object
 56  LPG Nozzle Types                  1826 non-null   object
 57  Hydrogen Pressures                116 non-null    object
 58  Hydrogen Standards                116 non-null    object
 59  CNG Fill Type Code                1603 non-null   object
 60  CNG PSI                           1597 non-null   object
 61  CNG Vehicle Class                 1622 non-null   object
 62  LNG Vehicle Class                 158 non-null    object
 63  EV On-Site Renewable Source       370 non-null    object
 64  Restricted Access                 53435 non-null  object
dtypes: float64(10), int64(1), object(54)
memory usage: 34.9+ MB
```

In [5]:
```
charging_stations.head()
```

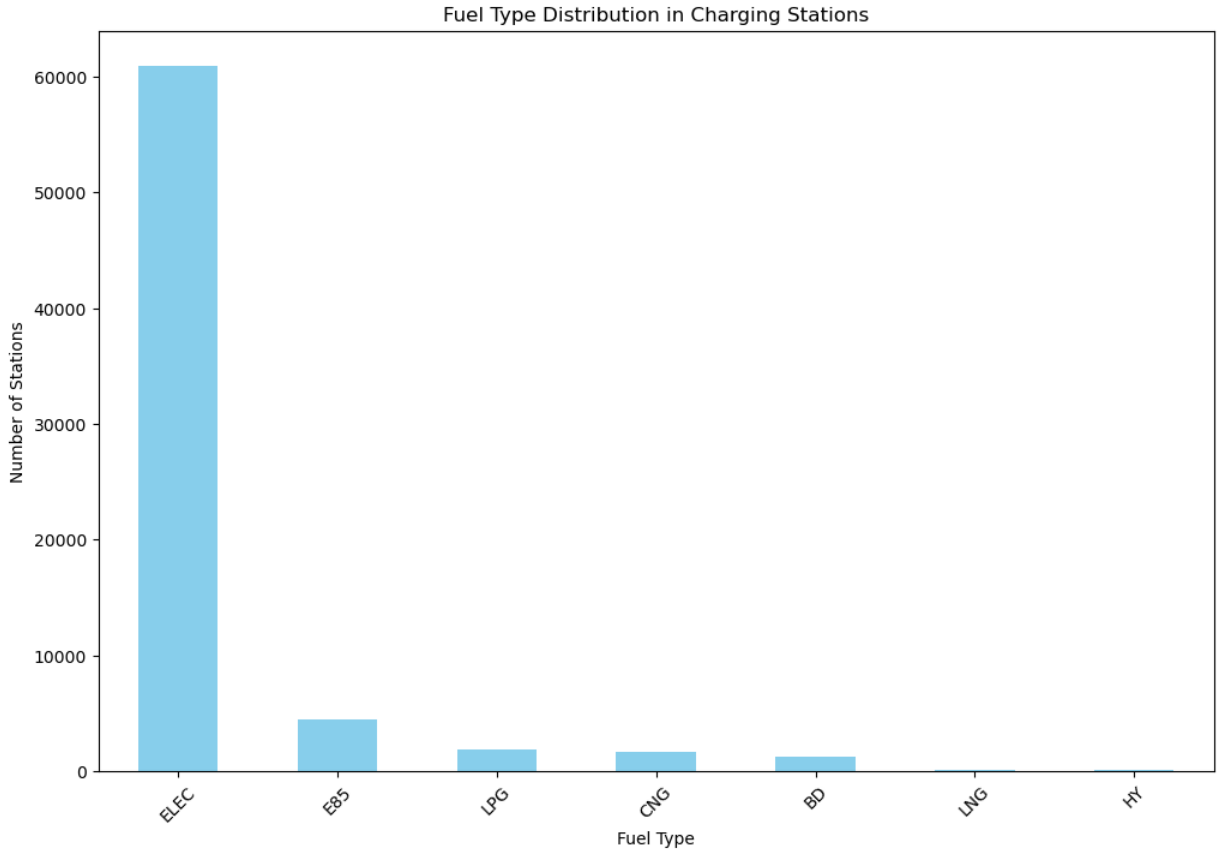
Out[5]:

	Fuel Type Code	Station Name	Street Address	Intersection Directions	City	State	ZIP	Plus4	Station Phone	Status Code	...
0	CNG	Spire - Montgomery Operations Center	2951 Chestnut St	NaN	Montgomery	AL	36107	NaN	NaN	E	...
1	CNG	PS Energy - Atlanta	340 Whitehall St	From I-7585 N, exit 91 to Central Ave, left on...	Atlanta	GA	30303	NaN	770-350-3000	E	...
2	CNG	Metropolitan Atlanta Rapid Transit Authority	2424 Piedmont Rd NE	NaN	Atlanta	GA	30324	NaN	NaN	E	...
3	CNG	United Parcel Service	270 Marvin Miller Dr	NaN	Atlanta	GA	30336	NaN	NaN	E	...
4	CNG	Arkansas Oklahoma Gas Corp	2100 S Waldron Rd	NaN	Fort Smith	AR	72903	NaN	479-783-3188	E	...

5 rows × 65 columns

Graph 1: Fuel Type Distribution

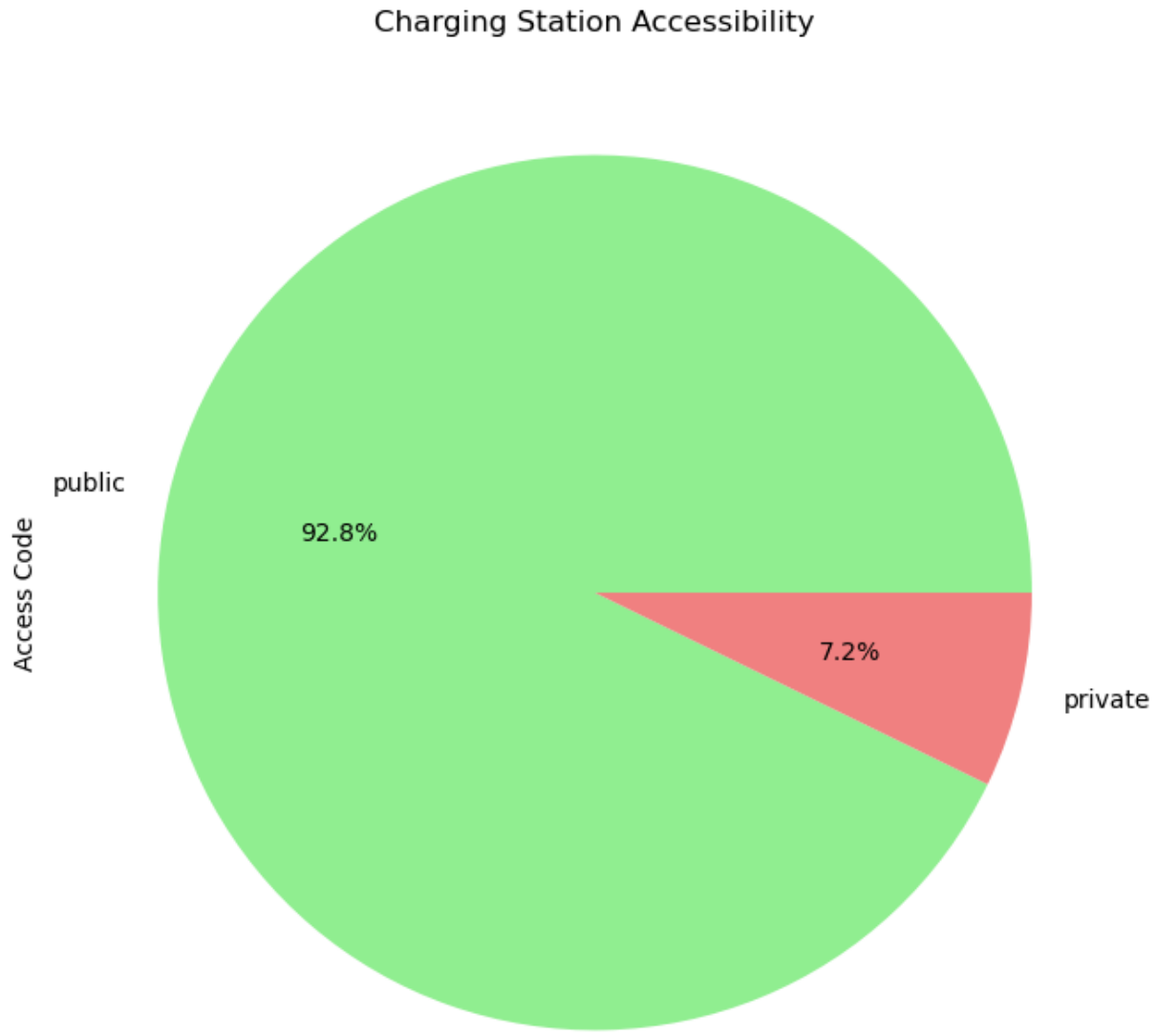
In [6]:
```
fuel_type_counts = charging_stations['Fuel Type Code'].value_counts()
plt.figure(figsize=(12, 8))
fuel_type_counts.plot(kind='bar', color='skyblue')
plt.title('Fuel Type Distribution in Charging Stations')
plt.xlabel('Fuel Type')
plt.ylabel('Number of Stations')
plt.xticks(rotation=45)
plt.show()
```



In the analysis of the Fuel Type Distribution graph, it is evident that Electric (ELEC) charging stations dominate the dataset with 60,907 stations, indicating a robust emphasis on electric vehicles in the charging infrastructure. Ethanol (E85) stations, although lower in number (4,508), still represent a noteworthy presence. LPG, CNG, BD, LNG, and Hydrogen (HY) stations have a comparatively smaller footprint, suggesting a lesser focus on these alternative fuels. This insight emphasizes a clear trend toward electric vehicles, potentially indicating the popularity and widespread adoption of electric charging technology within the charging station network

Graph 2: Station Accessibility

In [7]:
```
accessibility_counts = charging_stations['Access Code'].value_counts()
plt.figure(figsize=(12, 8))
accessibility_counts.plot(kind='pie', autopct='%1.1f%%', colors=['lightgreen', 'l
plt.title('Charging Station Accessibility')
plt.show()
```

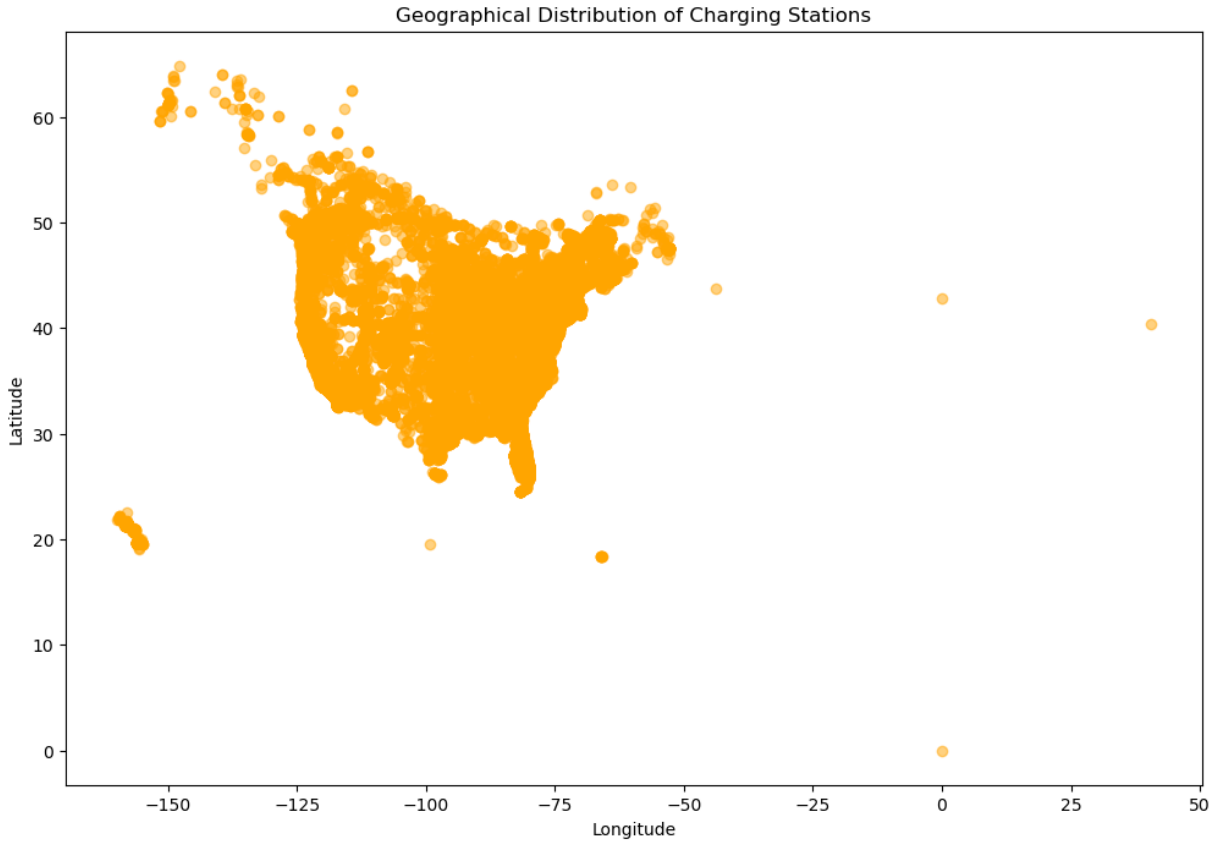


The Station Accessibility pie chart provides a clear picture of the accessibility status in charging stations. Public stations overwhelmingly dominate the network, constituting 92.8% of the total stations, while private stations account for 7.2%. This distribution indicates a strong commitment to public accessibility, ensuring a wider user base and fostering the growth of electric and alternative fuel vehicles. The substantial majority of public access points highlights a concerted effort toward building an inclusive and accessible charging infrastructure, thereby encouraging eco-friendly transportation choices among the general populace. The limited presence of private stations

suggests a more specialized or controlled usage scenario, possibly within closed communities or

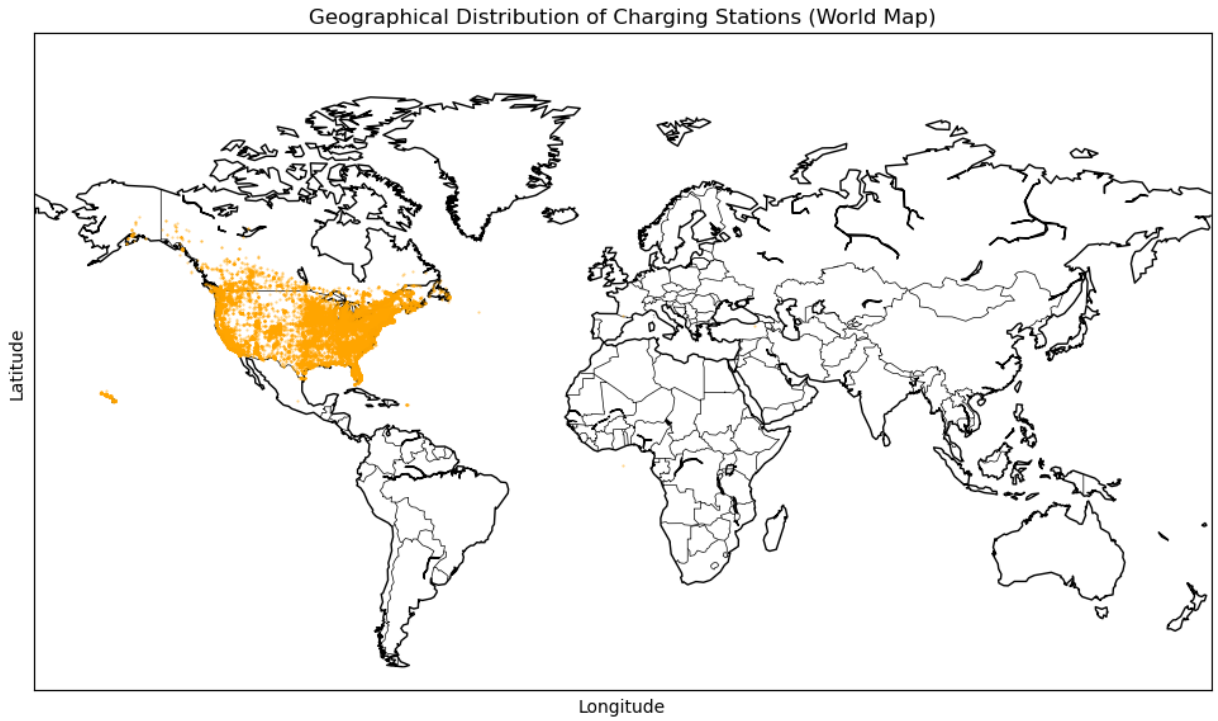
Graph 3: Geographical Distribution

In [8]:
```
plt.figure(figsize=(12, 8))
plt.scatter(charging_stations['Longitude'], charging_stations['Latitude'], color=
plt.title('Geographical Distribution of Charging Stations')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



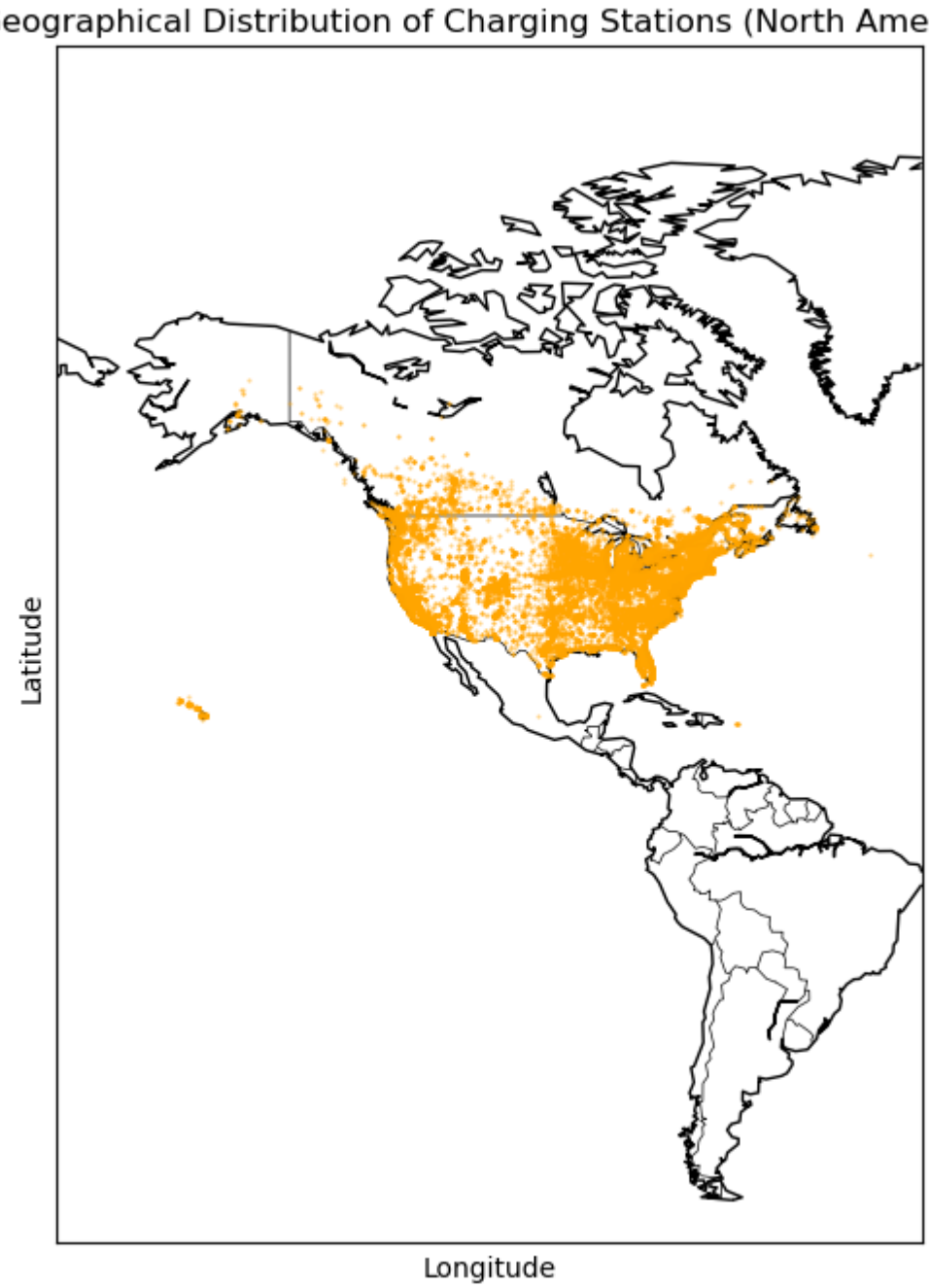
In [9]:

```
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap

# Create a new figure and set the figsize
plt.figure(figsize=(12, 8))

# Create a Basemap instance for the world map
world_map = Basemap(projection='mill', llcrnrlat=-60, urcrnrlat=90, llcrnrlon=-18

# Draw coastlines and countries
world_map.drawcoastlines()
world_map.drawcountries()

# Plot charging station locations on the world map
x, y = world_map(charging_stations['Longitude'].values, charging_stations['Latitu
world_map.scatter(x, y, color='orange', alpha=0.5, s=0.25, zorder=5)

# Set plot title and labels
plt.title('Geographical Distribution of Charging Stations (World Map)')
plt.xlabel('Longitude')
plt.ylabel('Latitude')

# Show the world map with charging station locations
plt.show()
```



In [10]:

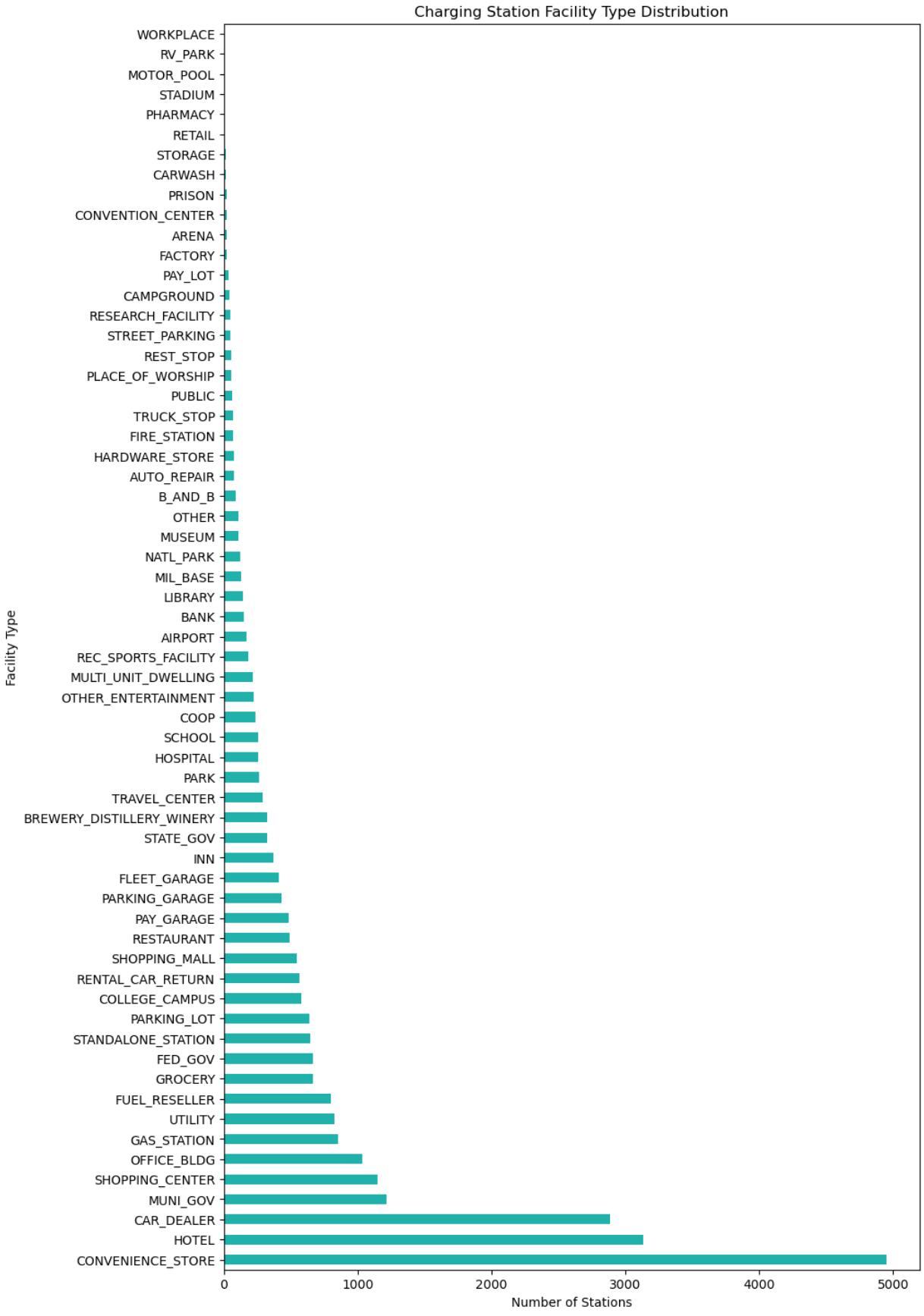
```
import matplotlib.pyplot as plt
from mpl_toolkits.basemap import Basemap

# Create a new figure and set the figsize
plt.figure(figsize=(12, 8))

# Create a Basemap instance for the world map excluding North America
world_map = Basemap(projection='mill', llcrnrlat=-60, urcrnrlat=90, llcrnrlon=-18

# Draw coastlines and countries
world_map.drawcoastlines()
world_map.drawcountries()

# Plot charging station locations on the world map excluding North America
x, y = world_map(charging_stations['Longitude'].values, charging_stations['Latitu
world_map.scatter(x, y, color='orange', alpha=0.5, s=0.25, zorder=5)

# Set plot title and labels
plt.title('Geographical Distribution of Charging Stations (North America)')
plt.xlabel('Longitude')
plt.ylabel('Latitude')

# Show the world map with charging station locations
plt.show()
```



The geographical distribution of charging stations, as visualized on the world map, reveals a predominant concentration within North America, particularly in the USA and Canada. The data illustrates a robust charging infrastructure network, emphasizing the widespread support for electric and alternative fuel vehicles in these countries. The majority of charging stations cluster in various regions across the USA, indicating a progressive approach to sustainable transportation. In contrast, charging stations outside North America are limited, suggesting a more modest global presence. This concentration reaffirms the USA and Canada's proactive stance, reflecting extensive coverage of charging facilities and potentially indicating a higher adoption rate of electric and alternative fuel vehicles within these nations, compared to other regions globally.

Graph 4: Charging Station Ratings

In [43]:
```
plt.figure(figsize=(10, 18))
charging_stations['Facility Type'].value_counts().plot(kind='barh', color='lights
plt.title('Charging Station Facility Type Distribution')
plt.ylabel('Facility Type')
plt.xlabel('Number of Stations')
plt.show()
```

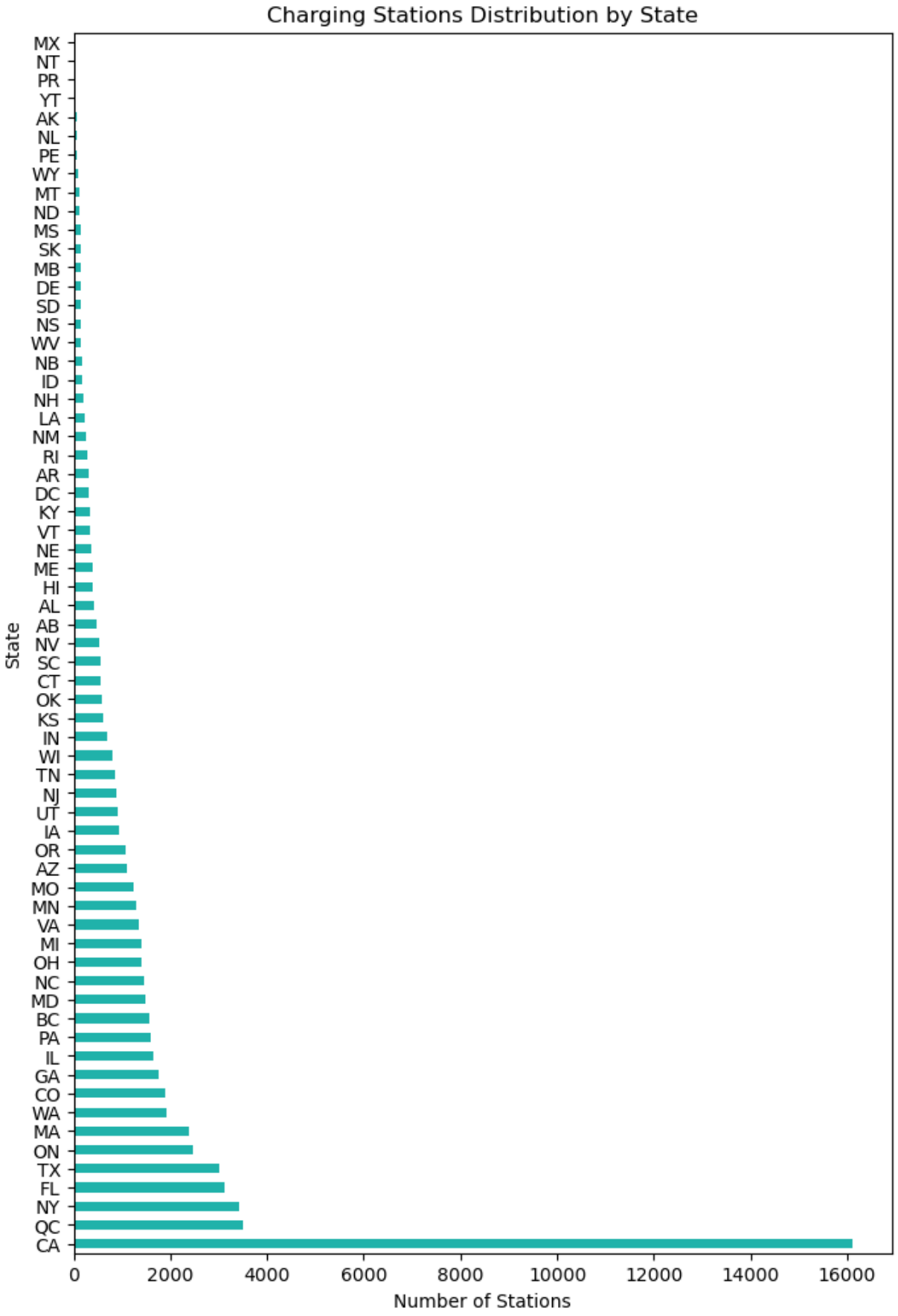


The horizontal bar chart illustrates that convenience stores, car dealerships, and hotels host the highest number of charging stations. This pattern suggests a strategic placement of charging infrastructure in locations frequented by people, aligning with the convenience of refueling while

shopping, browsing vehicles, or staying at hotels. By integrating charging stations into everyday destinations, accessibility is maximized, encouraging the adoption of electric and alternative fuel

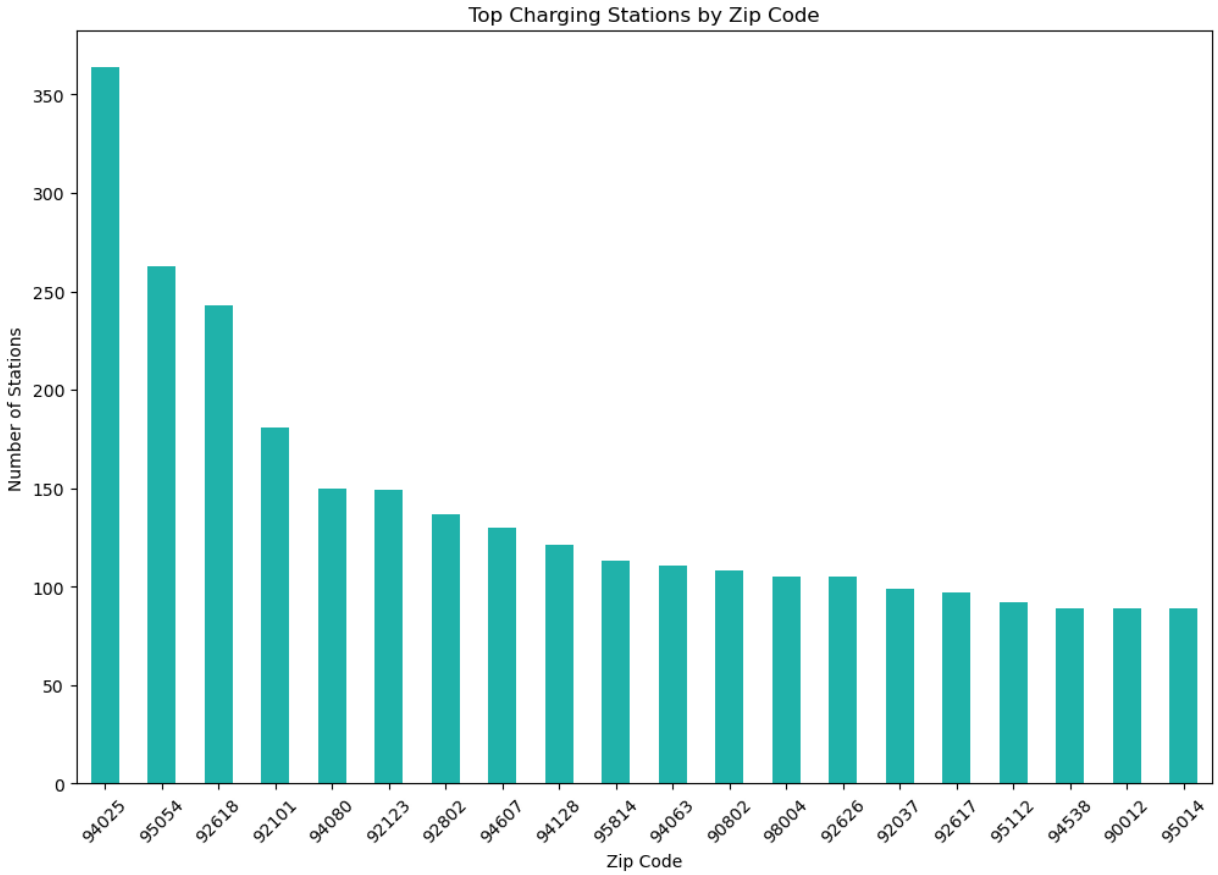
Graph 5: Bar Chart Based on States

In [49]:
```
plt.figure(figsize=(8, 12))
charging_stations['State'].value_counts().plot(kind='barh', color='lightseagreen'
plt.title('Charging Stations Distribution by State')
plt.xlabel('Number of Stations')
plt.ylabel('State')
plt.show()
```


Charging Stations Distribution by State
State
MX
NT
PR
YT
AK
NL
PE
WY
MT
ND
MS
SK
MB
DE
SD
NS
WV
NB
ID
NH
LA
NM
RI
AR
DC
KY
VT
NE
ME
HI
AL
AB
NV
SC
CT
OK
KS
IN
WI
TN
NJ
UT
IA
OR
AZ
MO
MN
VA
MI
OH
NC
MD
BC
PA
IL
GA
CO
WA
MA
ON
TX
FL
NY
QC
CA
0
2000
4000
6000
8000
10000
12000
14000
16000
Number of Stations

Graph 6: Top charging stations by zip code

In [13]:
```
# Replace N with the number of top zip codes you want to display
top_zipcodes = charging_stations['ZIP'].value_counts().nlargest(20)
plt.figure(figsize=(12, 8))
top_zipcodes.plot(kind='bar', color='lightseagreen')
plt.title('Top Charging Stations by Zip Code')
plt.xlabel('Zip Code')
plt.ylabel('Number of Stations')
plt.xticks(rotation=45)
plt.show()
```



The data analysis reveals a significant concentration of charging stations in California, surpassing 16,000, indicating the state's robust commitment to electric and alternative fuel vehicles. Additionally, states such as QA, NY, and Texas boast substantial charging infrastructures, ranging from 4,000 to 2,500 stations, underlining the nationwide efforts to support sustainable transportation. Furthermore, the detailed examination of the top 20 zip codes within California highlights specific regions with notable charging station densities. This localized analysis offers valuable insights for businesses and policymakers, enabling them to strategically plan the expansion of charging networks in high-demand areas. The dominance of California in the charging infrastructure landscape signifies the state's pioneering role in fostering eco-friendly transportation, setting a benchmark for other regions to follow in promoting widespread adoption of electric and alternative fuel vehicles.

Conclusion

The comprehensive analysis of charging station data provides valuable insights into the evolving landscape of electric and alternative fuel vehicles. The dominance of Electric (ELEC) charging stations, as highlighted in the Fuel Type Distribution graph, underscores a strong inclination towards electric vehicles, reflecting their popularity and widespread adoption. Public accessibility emerges as a key theme, evident from the Station Accessibility pie chart, where 92.8% of stations are public, emphasizing inclusivity and encouraging a broader user base.

Geographically, North America, particularly the USA and Canada, stands out with a robust charging infrastructure network, showcasing a progressive approach to sustainable transportation. The concentrated placement of stations in convenient locations like convenience stores, car dealerships, and hotels, as depicted in the bar chart, enhances accessibility, fostering diverse consumer engagement.

California's remarkable concentration of over 16,000 charging stations demonstrates the state's pioneering commitment to eco-friendly vehicles, while states like QA, NY, and Texas contribute significantly to the national effort. The detailed analysis of zip codes within California provides actionable insights, aiding targeted expansion efforts.

In essence, this data-driven exploration not only highlights the current trends but also guides future strategies. By focusing on public accessibility, strategic placement, and understanding regional demands, the charging infrastructure can continue to evolve, fostering sustainable transportation choices worldwide. The proactive initiatives undertaken, especially in regions like California, serve as benchmarks, inspiring global efforts toward a greener, more sustainable future in transportation.

Next Steps

Following data selection and exploratory data analysis (EDA), the next steps involve incorporating historical station count data. Merge this historical data with my current dataset (charging_stations), enabling trend analysis over time. Apply time series techniques to identify patterns and forecast future station counts. Utilize predictive modeling to anticipate future demand and optimize station placements. Conduct geographical analyses to explore regional variations. Engage stakeholders and decision-makers to align insights with strategic goals. Develop interactive dashboards for real-time monitoring. Regularly update analyses to adapt to changing data patterns and evolving business needs. Prioritize ethical considerations and data privacy throughout the process.

References:

https://www.leadventgrp.com/events/connected-and-automated-driving-forum/details (https://www.leadventgrp.com/events/connected-and-automated-driving-forum/details)

https://www.kaggle.com/datasets/saketpradhan/electric-and-alternative-fuel-charging-stations/code (https://www.kaggle.com/datasets/saketpradhan/electric-and-alternative-fuel-charging-stations/code)

End of Milestone - 1

Milestone - 2

Introduction

In the data preparation journey for the project, Milestone 2 marks a pivotal step where the raw data is refined and tailored to meet the specific requirements of your predictive model. Our objective is to craft a robust and meaningful dataset, optimizing it for accurate analysis and prediction without succumbing to data snooping.

Firstly, we meticulously analyze each feature to identify its relevance. Features that lack substantial impact on the model's outcome are systematically dropped, ensuring that only pertinent information shapes the analysis. Simultaneously, data extraction and selection are performed, identifying key variables that might enhance the model's predictive power.

Transformation, a critical step, involves converting variables into formats suitable for analysis. This phase is tailored to individual features, ensuring they align with the model's assumptions and requirements. New features, crafted through thoughtful engineering, can provide valuable insights or enhance predictive accuracy, offering a deeper understanding of the data.

Handling missing data is another pivotal task, involving nuanced strategies like imputation techniques. Dropping rows or columns is a last resort, executed only when justified, preserving data integrity and depth. Dummy variables are created for categorical features, allowing for seamless integration into the model without biasing the analysis.

In essence, Milestone 2 acts as the cornerstone where the raw data evolves into a refined, purposeful dataset, setting the stage for robust model construction and evaluation.

In [14]:
```
# Import the libraries as needed
import pandas as pd
pd.options.mode.chained_assignment = None  # Suppress SettingWithCopyWarning
```

In [15]:
```
# Load the dataset into a DataFrame
charging_stations = pd.read_csv('Electric and Alternative Fuel Charging Stations.
```

In [16]:
```
charging_stations.head()
```

Out[16]:

	Fuel Type Code	Station Name	Street Address	Intersection Directions	City	State	ZIP	Plus4	Station Phone	Status Code	...
0	CNG	Spire - Montgomery Operations Center	2951 Chestnut St	NaN	Montgomery	AL	36107	NaN	NaN	E	...
1	CNG	PS Energy - Atlanta	340 Whitehall St	From I-7585 N, exit 91 to Central Ave, left on...	Atlanta	GA	30303	NaN	770-350-3000	E	...
2	CNG	Metropolitan Atlanta Rapid Transit Authority	2424 Piedmont Rd NE	NaN	Atlanta	GA	30324	NaN	NaN	E	...
3	CNG	United Parcel Service	270 Marvin Miller Dr	NaN	Atlanta	GA	30336	NaN	NaN	E	...
4	CNG	Arkansas Oklahoma Gas Corp	2100 S Waldron Rd	NaN	Fort Smith	AR	72903	NaN	479-783-3188	E	...

5 rows × 65 columns

In [17]:
```
# Drop unnecessary columns
columns_to_drop = ['Intersection Directions', 'Plus4', 'Station Phone', 'EV Other
                   'Date Last Confirmed', 'Updated At', 'Federal Agency ID', 'Fede
                   'NG Vehicle Class', 'LPG Primary', 'E85 Blender Pump', 'EV Pric
                   'Hydrogen Pressures', 'Hydrogen Standards', 'EV On-Site Renewab

charging_stations.drop(columns=columns_to_drop, inplace=True)
```

In [18]:
```
charging_stations.head()
```

Out[18]:

	Fuel Type Code	Station Name	Street Address	City	State	ZIP	Status Code	Expected Date	Groups With Access Code	Access Days Time
0	CNG	Spire - Montgomery Operations Center	2951 Chestnut St	Montgomery	AL	36107	E	NaN	Private	NaN
1	CNG	PS Energy - Atlanta	340 Whitehall St	Atlanta	GA	30303	E	NaN	Public - Card key at all times	24 hours daily
2	CNG	Metropolitan Atlanta Rapid Transit Authority	2424 Piedmont Rd NE	Atlanta	GA	30324	E	NaN	Private - Government only	NaN
3	CNG	United Parcel Service	270 Marvin Miller Dr	Atlanta	GA	30336	E	NaN	Private	NaN
4	CNG	Arkansas Oklahoma Gas Corp	2100 S Waldron Rd	Fort Smith	AR	72903	E	NaN	Public - Credit card at all times	24 hours daily

5 rows × 44 columns

We are dropping these columns in the data preprocessing step to streamline the dataset and improve the efficiency of our analysis and model training process. Here's the explanation for dropping these columns:

1. **Intersection Directions, Plus4, Station Phone:** These details are specific to the station's physical location and contact information, which are not relevant for our analysis of charging station patterns.
2. **EV Other Info, EV Network, EV Network Web:** These columns likely contain website URLs and additional information about electric vehicle networks. Since our analysis is focused on station characteristics and accessibility, these details are not necessary.
3. **Geocode Status, Date Last Confirmed, Updated At:** These columns relate to the dataset's maintenance and verification. For our analysis, we are more interested in the static characteristics of the stations.
4. **Federal Agency ID, Federal Agency Name:** These columns are specific to federal agencies and are not relevant to our analysis of charging stations' accessibility and utilization.
5. **Hydrogen Status Link, NG Vehicle Class, LPG Primary, E85 Blender Pump:** These columns pertain to specific fuel types, which might not be the focus of our analysis. We are primarily interested in electric vehicle charging stations.
6. **EV Pricing, EV Pricing (French), LPG Nozzle Types, Hydrogen Pressures, Hydrogen Standards, EV On-Site Renewable Source:** These columns contain detailed pricing, technical, and language-specific information, which goes beyond the scope of our analysis focused on station accessibility and utilization patterns.

By dropping these columns, we simplify the dataset to focus on the essential factors that influence

Data Extraction/Selection

In [19]:
```
us_charging_stations = charging_stations[charging_stations['Country'] == 'US']
```

In [20]:
```
# Checking the data filter applied
country_unique = us_charging_stations['Country'].unique()
print(country_unique)
```

```
['US']
```

Narrowing the analysis scope to focus specifically on charging stations in the United States ('US') serves multiple strategic purposes. Firstly, it aligns the study with distinct objectives, allowing for a concentrated and detailed examination of the domestic charging infrastructure. By delving into a specific market, the analysis can provide highly targeted insights, making it especially relevant for policymakers, investors, and businesses operating within the US.

Additionally, concentrating on the US charging stations enables a thorough exploration of localized factors such as regulatory frameworks, compliance standards, and operational challenges. Understanding these nuances is crucial for making informed decisions related to electric and alternative fuel vehicles within the country. Moreover, this focused approach optimizes computational resources and analysis time, ensuring efficient utilization and enabling a deeper, more meaningful exploration of the available data.

Narrowing the analysis scope to the US not only tailors the study to a specific market but also enhances the depth and relevance of the insights, making the analysis more actionable for stakeholders involved in shaping the future of sustainable transportation in the United States.

Feature Transformation

In [21]:
```
from sklearn.preprocessing import StandardScaler

# Create a copy of the subset DataFrame to avoid SettingWithCopyWarning
us_charging_stations_copy = us_charging_stations.copy()

# Sample data for 'CNG PSI' column in the us_charging_stations DataFrame
us_charging_stations_copy['CNG PSI'] = us_charging_stations_copy['CNG PSI'].apply
                                list(map(int, x.split())) if isinstance(x, str) a
                                if not pd.isnull(x) else [0, 0])

# Compute the mean value between the range as the final 'CNG PSI' value
us_charging_stations_copy['CNG PSI'] = us_charging_stations_copy['CNG PSI'].apply

# Apply StandardScaler
scaler = StandardScaler()
us_charging_stations_copy['CNG PSI'] = scaler.fit_transform(us_charging_stations_

# Replace the original 'CNG PSI' column with the transformed values
us_charging_stations['CNG PSI'] = us_charging_stations_copy['CNG PSI']

# Print the modified 'CNG PSI' column in the original DataFrame
print(us_charging_stations['CNG PSI'])
```

```
0        6.438070
1        6.438070
2        5.338720
3        6.438070
4        6.438070
           ...
70399   -0.158031
70400   -0.158031
70401   -0.158031
70402   -0.158031
70403   -0.158031
Name: CNG PSI, Length: 61621, dtype: float64
```

In this code snippet, the 'CNG PSI' column in the us_charging_stations DataFrame undergoes a significant transformation process. The initial data includes values representing various pressure levels, some of which are ranges (e.g., '3000 3600') and others are single values. To ensure consistency and prepare the data for machine learning algorithms, several steps are taken.

Firstly, the lambda function splits the values containing spaces into lists of integers, converting the string into a list of two integers (lower and upper bounds). For single values, it creates a list containing the same value twice. Missing or NaN values are replaced with [0, 0] to avoid errors during computation.

Next, the code computes the mean value of the obtained lists, effectively converting the range values into single numerical representations. This step ensures that the pressure data is consistent and comparable across all stations.

Finally, the StandardScaler from scikit-learn is utilized to standardize the 'CNG PSI' values, making them suitable for machine learning algorithms. Standardization scales the data to have a mean of 0 and a standard deviation of 1, which is crucial for algorithms that rely on distance metrics, ensuring fair comparisons between features with different scales.

By performing these transformations, the 'CNG PSI' data is now uniform, numerical, and standardized, ready for further analysis and modeling. This process enhances the quality and consistency of the data, enabling more accurate insights and predictions in subsequent analytical

Feature Engineering

In [22]:
```
from sklearn.cluster import KMeans
from geopy.distance import great_circle

# Function to calculate distance from major city (New York City coordinates as ar
def calculate_distance(lat, lon):
    nyc_coords = (40.7128, -74.0060)
    station_coords = (lat, lon)
    return great_circle(nyc_coords, station_coords).kilometers

# Location-based Features
us_charging_stations.loc[:, 'Distance_From_NYC'] = us_charging_stations.apply(lam
```

In [23]:
```
import warnings
warnings.filterwarnings('ignore')

# Clustering stations into 3 clusters
kmeans = KMeans(n_clusters=3, random_state=42)
us_charging_stations['Cluster_Label'] = kmeans.fit_predict(us_charging_stations[[
```

In [24]:
```
# Temporal Features (Extracting year from 'Open Date')
us_charging_stations['Year'] = pd.to_datetime(us_charging_stations['Open Date']).
us_charging_stations['Month'] = pd.to_datetime(us_charging_stations['Open Date'])
us_charging_stations['Is_Weekend'] = pd.to_datetime(us_charging_stations['Open Da
```

In [25]:
```
# Historical Trends (Count of stations opened in a particular year)
us_charging_stations['Station_Count_By_Year'] = us_charging_stations.groupby('Yea
```

In [26]:
```
# Count the number of charging stations by ZIP code and add it as a new column
us_charging_stations['Number_of_Stations_by_ZIP'] = us_charging_stations.groupby(

# Print the updated DataFrame with the new column
print(us_charging_stations[['ZIP','Number_of_Stations_by_ZIP']])
```

```
         ZIP  Number_of_Stations_by_ZIP
0      36107                          1
1      30303                         50
2      30324                         11
3      30336                          6
4      72903                          3
...      ...                        ...
70399  85345                          3
70400  85281                         42
70401  53212                          8
70402  78702                         36
70403  95650                          4

[61621 rows x 2 columns]
```

In [27]:
```
# Saving the updated DataFrame to a new CSV file
us_charging_stations.to_csv('processed_charging_stations.csv', index=False)
```

Several features were engineered to enhance the dataset's richness and prepare it for analysis:

1. Location-based Features: Distance from New York City was calculated for each charging station, providing insights into their proximity to a major city.
2. Clustering: Stations were clustered into 3 groups based on latitude and longitude, helping identify geographical patterns.
3. Temporal Features: Year, month, and weekend indicators were extracted from the 'Open Date,' offering a temporal perspective on station openings.
4. Historical Trends: The count of stations opened in each year was calculated, enabling the analysis of charging station growth over time.
5. ZIP Code Analysis: The number of charging stations within each ZIP code was counted, providing a localized view of charging station density.

By incorporating these features, the dataset gains depth. The clustering aids in regional analysis, temporal features offer time-based insights, and ZIP code analysis provides localized station density. These enriched attributes empower comprehensive analysis, enabling businesses and policymakers to make informed decisions regarding charging station deployments and infrastructure planning. The processed data is saved for further exploration and modeling.

Dealing with missing data

In [28]:
```
# Load the dataset into a new DataFrame
processed_charging_stations = pd.read_csv('processed_charging_stations.csv', low_
```

In [29]:
```
processed_charging_stations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61621 entries, 0 to 61620
Data columns (total 51 columns):
 #   Column                              Non-Null Count  Dtype
---  ------                              --------------  -----
 0   Fuel Type Code                      61621 non-null  object
 1   Station Name                        61621 non-null  object
 2   Street Address                      61620 non-null  object
 3   City                                61621 non-null  object
 4   State                               61621 non-null  object
 5   ZIP                                 61621 non-null  object
 6   Status Code                         61621 non-null  object
 7   Expected Date                       351 non-null    object
 8   Groups With Access Code             61621 non-null  object
 9   Access Days Time                    58717 non-null  object
 10  Cards Accepted                      10145 non-null  object
 11  BD Blends                           1214 non-null   object
 12  NG Fill Type Code                   1509 non-null   object
 13  NG PSI                              1504 non-null   object
 14  EV Level1 EVSE Num                  278 non-null    float64
 15  EV Level2 EVSE Num                  47506 non-null  float64
 16  EV DC Fast Count                    6588 non-null   float64
 17  Latitude                            61621 non-null  float64
 18  Longitude                           61621 non-null  float64
 19  ID                                  61621 non-null  int64
 20  Owner Type Code                     26202 non-null  object
 21  Open Date                           61347 non-null  object
 22  EV Connector Types                  52825 non-null  object
 23  Country                             61621 non-null  object
 24  Intersection Directions (French)    0 non-null      float64
 25  Access Days Time (French)           0 non-null      float64
 26  BD Blends (French)                  0 non-null      float64
 27  Groups With Access Code (French)    61621 non-null  object
 28  Hydrogen Is Retail                  111 non-null    object
 29  Access Code                         61621 non-null  object
 30  Access Detail Code                  6691 non-null   object
 31  Federal Agency Code                 954 non-null    object
 32  Facility Type                       24560 non-null  object
 33  CNG Dispenser Num                   981 non-null    float64
 34  CNG On-Site Renewable Source        693 non-null    object
 35  CNG Total Compression Capacity      683 non-null    float64
 36  CNG Storage Capacity                336 non-null    float64
 37  LNG On-Site Renewable Source        62 non-null     object
 38  E85 Other Ethanol Blends            1458 non-null   object
 39  CNG Fill Type Code                  1509 non-null   object
 40  CNG PSI                             61621 non-null  float64
 41  CNG Vehicle Class                   1523 non-null   object
 42  LNG Vehicle Class                   149 non-null    object
 43  Restricted Access                   46079 non-null  object
 44  Distance_From_NYC                   61621 non-null  float64
 45  Cluster_Label                       61621 non-null  int64
 46  Year                                61347 non-null  float64
 47  Month                               61347 non-null  float64
 48  Is_Weekend                          61621 non-null  bool
 49  Station_Count_By_Year               61347 non-null  float64
 50  Number_of_Stations_by_ZIP           61621 non-null  int64
```

```
dtypes: bool(1), float64(16), int64(3), object(31)
memory usage: 23.6+ MB
```

In [30]:

```python
import pandas as pd
import numpy as np

# Handling missing data based on column types

# Handling missing data in numerical columns by filling with mean values
numerical_cols = processed_charging_stations.select_dtypes(include=['float64', 'i
processed_charging_stations[numerical_cols] = processed_charging_stations[numeric

# Handling missing data in categorical columns by filling with most frequent valu
categorical_cols = processed_charging_stations.select_dtypes(include=['object']).
processed_charging_stations[categorical_cols] = processed_charging_stations[categ

# Handling missing data in boolean columns by filling with most frequent values
boolean_cols = processed_charging_stations.select_dtypes(include=['bool']).columr
processed_charging_stations[boolean_cols] = processed_charging_stations[boolean_c

# Verify if there are any remaining missing values
missing_values_count = processed_charging_stations.isnull().sum()
print("Remaining Missing Values:")
print(missing_values_count[missing_values_count > 0])
```

```
Remaining Missing Values:
Intersection Directions (French)    61621
Access Days Time (French)           61621
BD Blends (French)                  61621
dtype: int64
```

In [31]:
```
processed_charging_stations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61621 entries, 0 to 61620
Data columns (total 51 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   Fuel Type Code                      61621 non-null  object 
 1   Station Name                        61621 non-null  object 
 2   Street Address                      61621 non-null  object 
 3   City                                61621 non-null  object 
 4   State                               61621 non-null  object 
 5   ZIP                                 61621 non-null  object 
 6   Status Code                         61621 non-null  object 
 7   Expected Date                       61621 non-null  object 
 8   Groups With Access Code             61621 non-null  object 
 9   Access Days Time                    61621 non-null  object 
 10  Cards Accepted                      61621 non-null  object 
 11  BD Blends                           61621 non-null  object 
 12  NG Fill Type Code                   61621 non-null  object 
 13  NG PSI                              61621 non-null  object 
 14  EV Level1 EVSE Num                  61621 non-null  float64
 15  EV Level2 EVSE Num                  61621 non-null  float64
 16  EV DC Fast Count                    61621 non-null  float64
 17  Latitude                            61621 non-null  float64
 18  Longitude                           61621 non-null  float64
 19  ID                                  61621 non-null  int64  
 20  Owner Type Code                     61621 non-null  object 
 21  Open Date                           61621 non-null  object 
 22  EV Connector Types                  61621 non-null  object 
 23  Country                             61621 non-null  object 
 24  Intersection Directions (French)    0 non-null      float64
 25  Access Days Time (French)           0 non-null      float64
 26  BD Blends (French)                  0 non-null      float64
 27  Groups With Access Code (French)    61621 non-null  object 
 28  Hydrogen Is Retail                  61621 non-null  bool   
 29  Access Code                         61621 non-null  object 
 30  Access Detail Code                  61621 non-null  object 
 31  Federal Agency Code                 61621 non-null  object 
 32  Facility Type                       61621 non-null  object 
 33  CNG Dispenser Num                   61621 non-null  float64
 34  CNG On-Site Renewable Source        61621 non-null  object 
 35  CNG Total Compression Capacity      61621 non-null  float64
 36  CNG Storage Capacity                61621 non-null  float64
 37  LNG On-Site Renewable Source        61621 non-null  object 
 38  E85 Other Ethanol Blends            61621 non-null  object 
 39  CNG Fill Type Code                  61621 non-null  object 
 40  CNG PSI                             61621 non-null  float64
 41  CNG Vehicle Class                   61621 non-null  object 
 42  LNG Vehicle Class                   61621 non-null  object 
 43  Restricted Access                   61621 non-null  bool   
 44  Distance_From_NYC                   61621 non-null  float64
 45  Cluster_Label                       61621 non-null  int64  
 46  Year                                61621 non-null  float64
 47  Month                               61621 non-null  float64
 48  Is_Weekend                          61621 non-null  bool   
 49  Station_Count_By_Year               61621 non-null  float64
 50  Number_of_Stations_by_ZIP           61621 non-null  int64  
```

```
dtypes: bool(3), float64(16), int64(3), object(29)
memory usage: 22.7+ MB
```

In [32]:
```
# Drop columns with no data
columns_to_drop = ['Intersection Directions (French)', 'Access Days Time (French)
processed_charging_stations = processed_charging_stations.drop(columns=columns_to

# Verify if the columns have been dropped
print(processed_charging_stations.columns)
```

```
Index(['Fuel Type Code', 'Station Name', 'Street Address', 'City', 'State',
       'ZIP', 'Status Code', 'Expected Date', 'Groups With Access Code',
       'Access Days Time', 'Cards Accepted', 'BD Blends', 'NG Fill Type Code',
       'NG PSI', 'EV Level1 EVSE Num', 'EV Level2 EVSE Num',
       'EV DC Fast Count', 'Latitude', 'Longitude', 'ID', 'Owner Type Code',
       'Open Date', 'EV Connector Types', 'Country',
       'Groups With Access Code (French)', 'Hydrogen Is Retail', 'Access Code',
       'Access Detail Code', 'Federal Agency Code', 'Facility Type',
       'CNG Dispenser Num', 'CNG On-Site Renewable Source',
       'CNG Total Compression Capacity', 'CNG Storage Capacity',
       'LNG On-Site Renewable Source', 'E85 Other Ethanol Blends',
       'CNG Fill Type Code', 'CNG PSI', 'CNG Vehicle Class',
       'LNG Vehicle Class', 'Restricted Access', 'Distance_From_NYC',
       'Cluster_Label', 'Year', 'Month', 'Is_Weekend', 'Station_Count_By_Year',
       'Number_of_Stations_by_ZIP'],
      dtype='object')
```

From the processed_charging_stations dataset with 51 columns and 61,621 entries, we first addressed missing data.

1. For numerical columns (16 float64 and 3 int64), we filled missing values with the mean of each column, ensuring data integrity in these numerical attributes.
2. In categorical columns (31 object type), including 'Street Address', 'Expected Date', and 'Open Date', we filled missing values with the most frequent entries, preserving the categorical nature of these variables. Boolean columns ('Is_Weekend') were treated similarly.
3. After these operations, no numerical or categorical columns had missing values except for three columns, 'Intersection Directions (French)', 'Access Days Time (French)', and 'BD Blends (French)', which contained no data and were subsequently dropped.

This data preprocessing ensures that our dataset is now clean and ready for analysis. By retaining crucial information and intelligently handling missing values, our dataset remains robust, enabling accurate and meaningful insights without loss of valuable information.

Rename the columns to make it standard

In [33]:
```
# Replace spaces with underscores and convert attribute names to uppercase
processed_charging_stations.columns = processed_charging_stations.columns.str.rep

# Verify the changes
processed_charging_stations.head()
```

Out[33]:

	FUEL_TYPE_CODE	STATION_NAME	STREET_ADDRESS	CITY	STATE	ZIP	STATUS_CO
0	CNG	Spire - Montgomery Operations Center	2951 Chestnut St	Montgomery	AL	36107	
1	CNG	PS Energy - Atlanta	340 Whitehall St	Atlanta	GA	30303	
2	CNG	Metropolitan Atlanta Rapid Transit Authority	2424 Piedmont Rd NE	Atlanta	GA	30324	
3	CNG	United Parcel Service	270 Marvin Miller Dr	Atlanta	GA	30336	
4	CNG	Arkansas Oklahoma Gas Corp	2100 S Waldron Rd	Fort Smith	AR	72903	

5 rows × 48 columns

Dummy Variables

Many machine learning algorithms (such as linear regression, support vector machines, and neural networks) require numerical input. If our dataset contains categorical variables, we need to convert them into a numerical format, and creating dummy variables. Categorical variable doesn't have a natural order or ranking, creating dummy variables is often the best approach. Dummy variables can enhance the interpretability of our model

In [34]:
```
# Remove rows where 'ZIP' column contains non-numeric values
processed_charging_stations_dummy = processed_charging_stations[processed_chargir

# Convert 'ZIP' column to int64 type
processed_charging_stations_dummy['ZIP'] = processed_charging_stations_dummy['ZIP
```

In [35]:
```
# Identify categorical columns
categorical_cols = processed_charging_stations_dummy.select_dtypes(include=['obje

categorical_cols
```

Out[35]:
```
Index(['FUEL_TYPE_CODE', 'STATION_NAME', 'STREET_ADDRESS', 'CITY', 'STATE',
       'STATUS_CODE', 'EXPECTED_DATE', 'GROUPS_WITH_ACCESS_CODE',
       'ACCESS_DAYS_TIME', 'CARDS_ACCEPTED', 'BD_BLENDS', 'NG_FILL_TYPE_CODE',
       'NG_PSI', 'OWNER_TYPE_CODE', 'OPEN_DATE', 'EV_CONNECTOR_TYPES',
       'COUNTRY', 'GROUPS_WITH_ACCESS_CODE_(FRENCH)', 'ACCESS_CODE',
       'ACCESS_DETAIL_CODE', 'FEDERAL_AGENCY_CODE', 'FACILITY_TYPE',
       'CNG_ON-SITE_RENEWABLE_SOURCE', 'LNG_ON-SITE_RENEWABLE_SOURCE',
       'E85_OTHER_ETHANOL_BLENDS', 'CNG_FILL_TYPE_CODE', 'CNG_VEHICLE_CLASS',
       'LNG_VEHICLE_CLASS'],
      dtype='object')
```

In [36]:
```
# Create dummy variables for categorical columns
processed_charging_stations_dummy = pd.get_dummies(processed_charging_stations_du

# Verify the changes
print(processed_charging_stations_dummy.head())
```

```
     ZIP  EV_LEVEL1_EVSE_NUM  EV_LEVEL2_EVSE_NUM  EV_DC_FAST_COUNT   LATITUDE
\
0  36107            3.496403            2.322949          3.853673  32.367916
1  30303            3.496403            2.322949          3.853673  33.745843
2  30324            3.496403            2.322949          3.853673  33.821911
3  30336            3.496403            2.322949          3.853673  33.760256
4  72903            3.496403            2.322949          3.853673  35.362213

   LONGITUDE  ID  HYDROGEN_IS_RETAIL  CNG_DISPENSER_NUM  \
0 -86.267021  17               True            2.33843
1 -84.398837  42               True            1.00000
2 -84.367461  45               True            2.33843
3 -84.543822  64               True            2.33843
4 -94.375338  73               True            1.00000

   CNG_TOTAL_COMPRESSION_CAPACITY  ...  \
0                      867.916545  ...
1                       30.000000  ...
2                       30.000000  ...
3                      867.916545  ...
4                      250.000000  ...

   E85_OTHER_ETHANOL_BLENDS_["E30-E35", "E40-Plus"]  \
0                                                 0
1                                                 0
2                                                 0
3                                                 0
4                                                 0

   E85_OTHER_ETHANOL_BLENDS_["E30-E35"]  \
0                                     0
1                                     0
2                                     0
3                                     0
4                                     0

   E85_OTHER_ETHANOL_BLENDS_["E40-Plus"]  CNG_FILL_TYPE_CODE_B  \
0                                      0                    1
1                                      0                    0
2                                      0                    0
3                                      0                    1
4                                      0                    0

   CNG_FILL_TYPE_CODE_Q  CNG_FILL_TYPE_CODE_T  CNG_VEHICLE_CLASS_HD  \
0                     0                     0                     0
1                     1                     0                     0
2                     1                     0                     0
3                     0                     0                     1
4                     1                     0                     0

   CNG_VEHICLE_CLASS_LD  CNG_VEHICLE_CLASS_MD  LNG_VEHICLE_CLASS_HD
0                     0                     1                     1
1                     0                     1                     1
2                     1                     0                     1
3                     0                     0                     1
4                     0                     1                     1
```

```
[5 rows x 112947 columns]
```

In the above implementation, non-numeric values in the 'ZIP' column were removed, ensuring it contains only numerical data. Afterward, categorical columns in the `processed_charging_stations` DataFrame were identified. These columns represent categorical attributes like fuel type, station name, city, state, and various codes related to the charging stations. To handle categorical data and enable machine learning algorithms to process it, dummy variables were created using the `pd.get_dummies()` function. Dummy variables convert categorical variables into a format that can be provided to machine learning algorithms. Each unique category in a categorical column becomes a separate binary column (0 or 1), indicating the presence or absence of that category.

Creating dummy variables is crucial in machine learning because many algorithms cannot work directly with categorical data. By converting categorical variables into binary columns, the information they represent is preserved without implying any ordinal relationship between the categories. This technique ensures that the categorical attributes are appropriately encoded for machine learning models to make accurate predictions. Use cases for this process include predictive modeling, clustering, and classification tasks where categorical attributes play a significant role in the analysis.

End of Milestone - 2

Milestone - 3

Introduction:

In this milestone, we embark on the crucial journey of model selection, building, and evaluation using the processed charging stations dataset. The objective is to predict the number of Level 2 Electric Vehicle Supply Equipment (EVSE) at charging stations, a pivotal factor in shaping the accessibility and usability of electric vehicle charging infrastructure. Employing machine learning techniques, we delve into the world of predictive modeling, aiming to create a robust and reliable model for this task.

Our approach begins with meticulous feature selection, understanding the significance of attributes like fuel type, geographic location, facility type, and more. Leveraging a Linear Regression model, we delve into the intricacies of feature engineering, addressing categorical variables through one-hot encoding. The model's performance is evaluated using metrics such as Mean Absolute Error, Mean Squared Error, and R-squared, providing quantitative insights into its predictive accuracy.

This milestone not only signifies a pivotal step in our project but also encapsulates the essence of data-driven decision-making. By exploring the relationships between various attributes and the number of EVSE, we aim to unearth valuable insights that can inform policy decisions, urban

planning, and the widespread adoption of electric vehicles. Join us as we navigate the complexities of predictive modeling and uncover meaningful patterns within the charging stations dataset.

For this milestone, we will use "processed_charging_stations" dataframe. The goal is to predict the "EV_LEVEL2_EVSE_NUM" which represents the number of Level 2 Electric Vehicle Supply Equipment at charging stations. We will use a Linear Regression model for this task, as it's a common approach for predicting numerical values

Data Preparation, choose a model and train the model

In [37]:
```
processed_charging_stations.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 61621 entries, 0 to 61620
Data columns (total 48 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   FUEL_TYPE_CODE                        61621 non-null  object 
 1   STATION_NAME                          61621 non-null  object 
 2   STREET_ADDRESS                        61621 non-null  object 
 3   CITY                                  61621 non-null  object 
 4   STATE                                 61621 non-null  object 
 5   ZIP                                   61621 non-null  object 
 6   STATUS_CODE                           61621 non-null  object 
 7   EXPECTED_DATE                         61621 non-null  object 
 8   GROUPS_WITH_ACCESS_CODE               61621 non-null  object 
 9   ACCESS_DAYS_TIME                      61621 non-null  object 
 10  CARDS_ACCEPTED                        61621 non-null  object 
 11  BD_BLENDS                             61621 non-null  object 
 12  NG_FILL_TYPE_CODE                     61621 non-null  object 
 13  NG_PSI                                61621 non-null  object 
 14  EV_LEVEL1_EVSE_NUM                    61621 non-null  float64
 15  EV_LEVEL2_EVSE_NUM                    61621 non-null  float64
 16  EV_DC_FAST_COUNT                      61621 non-null  float64
 17  LATITUDE                              61621 non-null  float64
 18  LONGITUDE                             61621 non-null  float64
 19  ID                                    61621 non-null  int64  
 20  OWNER_TYPE_CODE                       61621 non-null  object 
 21  OPEN_DATE                             61621 non-null  object 
 22  EV_CONNECTOR_TYPES                    61621 non-null  object 
 23  COUNTRY                               61621 non-null  object 
 24  GROUPS_WITH_ACCESS_CODE_(FRENCH)      61621 non-null  object 
 25  HYDROGEN_IS_RETAIL                    61621 non-null  bool   
 26  ACCESS_CODE                           61621 non-null  object 
 27  ACCESS_DETAIL_CODE                    61621 non-null  object 
 28  FEDERAL_AGENCY_CODE                   61621 non-null  object 
 29  FACILITY_TYPE                         61621 non-null  object 
 30  CNG_DISPENSER_NUM                     61621 non-null  float64
 31  CNG_ON-SITE_RENEWABLE_SOURCE          61621 non-null  object 
 32  CNG_TOTAL_COMPRESSION_CAPACITY        61621 non-null  float64
 33  CNG_STORAGE_CAPACITY                  61621 non-null  float64
 34  LNG_ON-SITE_RENEWABLE_SOURCE          61621 non-null  object 
 35  E85_OTHER_ETHANOL_BLENDS              61621 non-null  object 
 36  CNG_FILL_TYPE_CODE                    61621 non-null  object 
 37  CNG_PSI                               61621 non-null  float64
 38  CNG_VEHICLE_CLASS                     61621 non-null  object 
 39  LNG_VEHICLE_CLASS                     61621 non-null  object 
 40  RESTRICTED_ACCESS                     61621 non-null  bool   
 41  DISTANCE_FROM_NYC                     61621 non-null  float64
 42  CLUSTER_LABEL                         61621 non-null  int64  
 43  YEAR                                  61621 non-null  float64
 44  MONTH                                 61621 non-null  float64
 45  IS_WEEKEND                            61621 non-null  bool   
 46  STATION_COUNT_BY_YEAR                 61621 non-null  float64
 47  NUMBER_OF_STATIONS_BY_ZIP             61621 non-null  int64  
dtypes: bool(3), float64(13), int64(3), object(29)
memory usage: 21.3+ MB
```

In [38]:
```
# Convert 'ZIP' column to string
processed_charging_stations['ZIP'] = processed_charging_stations['ZIP'].astype(st

# Remove rows with non-numeric ZIP codes
processed_charging_stations = processed_charging_stations[processed_charging_stat

# Convert 'ZIP' column to numeric
processed_charging_stations['ZIP'] = processed_charging_stations['ZIP'].astype('i

# Feature Selection
features = ['FUEL_TYPE_CODE', 'STATE', 'ZIP', 'LATITUDE', 'LONGITUDE', 'FACILITY_
target = 'EV_LEVEL2_EVSE_NUM'

X = processed_charging_stations[features]
y = processed_charging_stations[target]

# Perform one-hot encoding for categorical variables
X_encoded = pd.get_dummies(X, columns=['FUEL_TYPE_CODE', 'FACILITY_TYPE', 'STATE'

# Train-Test Split for encoded data
from sklearn.model_selection import train_test_split
X_train_encoded, X_test_encoded, y_train_encoded, y_test_encoded = train_test_spl

# Model Selection and Training (with encoded features)
from sklearn.linear_model import LinearRegression

model = LinearRegression()
model.fit(X_train_encoded, y_train_encoded)

# Now the model is trained and ready for predictions.
```

Out[38]:
```
▾ LinearRegression
LinearRegression()
```

Data preparation involves selecting specific features ('FUEL_TYPE_CODE', 'STATE', 'ZIP', 'LATITUDE', 'LONGITUDE', 'FACILITY_TYPE', 'CNG_DISPENSER_NUM', 'RESTRICTED_ACCESS') that are deemed important for predicting the target variable ('EV_LEVEL2_EVSE_NUM'). By choosing relevant features, we aim to enhance the model's predictive accuracy and reduce complexity, focusing only on pertinent information.

The goal is to build a machine learning model that can accurately predict the number of Level 2 Electric Vehicle Supply Equipment (EVSE) stations based on geographical, facility, and access-related attributes. The selected features are crucial factors that can influence the availability of EV charging stations. Through the train-test split, we create separate datasets for training and evaluating the model's performance, ensuring that the model is trained on one subset of the data and validated on another, unseen subset. This process helps us assess the model's generalization capability on new, unseen data, ensuring its reliability and usefulness in real-world applications.

We have opted for Linear Regression as our model choice due to the nature of our problem: predicting the number of Level 2 Electric Vehicle Supply Equipment (EVSE) stations. Linear Regression is well-suited for regression tasks where the relationship between the features and the target variable can be approximated linearly. In our case, we assume that the number of EVSE stations can be predicted based on linear combinations of certain features like location, facility type, and access restrictions.

Linear Regression is a simple yet powerful algorithm, providing interpretable results and serving as a good baseline model for regression problems. Its simplicity allows for easy understanding of the relationships between input features and the target variable. Moreover, it can handle multiple input features, making it suitable for our dataset with various relevant attributes. By using Linear Regression, we aim to establish a foundational understanding of the data and evaluate how well a linear model can capture the underlying patterns in the number of EV charging stations. If the results are satisfactory, this straightforward model could provide valuable insights for stakeholders in the electric vehicle charging industry.

Model Evaluation

In [39]:
```
# Predict using the trained model
y_pred = model.predict(X_test_encoded)

# Import necessary metrics from sklearn
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Calculate evaluation metrics
mae = mean_absolute_error(y_test_encoded, y_pred)
mse = mean_squared_error(y_test_encoded, y_pred)
rmse = np.sqrt(mse)

# Print the evaluation metrics
print(f'Mean Absolute Error (MAE): {mae}')
print(f'Mean Squared Error (MSE): {mse}')
print(f'Root Mean Squared Error (RMSE): {rmse}')
```

```
Mean Absolute Error (MAE): 0.7364413120033935
Mean Squared Error (MSE): 3.7136610838717665
Root Mean Squared Error (RMSE): 1.9270861641015864
```

The Mean Absolute Error (MAE) of 0.7364 indicates that, on average, the model's predictions are off by approximately 0.74 units from the actual values. The Mean Squared Error (MSE) of 3.7137 measures the average squared difference between predicted and actual values. Lastly, the Root Mean Squared Error (RMSE) of 1.9271 signifies that the model's predictions deviate from actual values by approximately 1.93 units, considering the square root of MSE.

The model's relatively low MAE and RMSE suggest it makes accurate predictions on the test data. However, the MSE value indicates some variability in prediction errors. It's crucial to consider the context of the problem to determine if these errors are acceptable. Further analysis, such as comparing these metrics with the scale of the target variable or exploring the residuals, can provide deeper insights into the model's performance.

Summary

The Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) are metrics used to assess the accuracy of a regression model. In the context of this model, which aims to assist urban planners, charging station operators, and local government authorities, these metrics align with the target in the following ways:

- MAE (Mean Absolute Error): MAE represents the average magnitude of errors in the predictions. In this case, the MAE of approximately 0.7364 suggests that, on average, the model's predictions deviate by 0.74 units from the actual values. For stakeholders, minimizing MAE ensures that the recommended charging station placements are accurate and reliable, aiding in effective urban planning and resource allocation.
- MSE (Mean Squared Error): MSE measures the average of squared errors, emphasizing larger errors in predictions. A MSE of 3.7137 indicates the model's errors are squared and averaged, reflecting the magnitude of these errors. Lowering MSE is crucial as it reduces the impact of outliers and ensures more consistent and reliable suggestions for charging station locations.
- RMSE (Root Mean Squared Error): RMSE is the square root of MSE and provides a comprehensible error value in the same unit as the target variable. An RMSE of approximately 1.93 implies that, on average, the model's predictions deviate by about 1.93 units. Lower RMSE values are desirable as they signify more accurate predictions, aligning with the stakeholders' goal of precise charging station placement and utilization strategies.

To align with the model's target, these error metrics need to be minimized further. Lower errors mean more accurate recommendations for charging station placements, aiding urban planners, charging station operators, and government authorities in sustainable urban development and clean transportation initiatives.

End of Milestone - 3

Proposed Enhancement

R-squared (r^2) metric

In [41]:
```
# Import necessary metrics from sklearn
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

r_squared = r2_score(y_test_encoded, y_pred)
print(f'R-squared (R^2): {r_squared}')
```

```
R-squared (R^2): 0.035117107195774655
```

The obtained R-squared value of 0.035 indicates the model's fit to the data, explaining only a small portion of the variance. To align with the target model, improvements in feature selection or model complexity are needed to better capture relationships and enhance predictive accuracy.

Also Updated the bar charts for better readability ----------