In [2]:
```python
import pandas as pd
import numpy as np
```

In [3]:
```python
# Reading the new CSV file
new_data = pd.read_csv('last_two_years_accidents.csv')
```

In [32]:
```python
new_data.shape
```

Out[32]:
```
(2009085, 42)
```

In [33]:
```python
# Checking for missing values
missing_values = new_data.isnull().sum()
print(missing_values)
```

```
ID                       0
Source                   0
Severity                 0
Start_Time               0
End_Time                 0
Start_Lat                0
Start_Lng                0
Distance(mi)             0
Street                   0
City                     0
County                   0
State                    0
Zipcode                  0
Country                  0
Timezone                 0
Airport_Code             0
Weather_Timestamp        0
Temperature(F)           0
Humidity(%)              0
Pressure(in)             0
Visibility(mi)           0
Wind_Direction           0
Wind_Speed(mph)          0
Weather_Condition        0
Amenity                  0
Bump                     0
Crossing                 0
Give_Way                 0
Junction                 0
No_Exit                  0
Railway                  0
Roundabout               0
Station                  0
Stop                     0
Traffic_Calming          0
Traffic_Signal           0
Turning_Loop             0
Sunrise_Sunset           0
Civil_Twilight           0
Nautical_Twilight        0
Astronomical_Twilight    0
Cluster                  0
dtype: int64
```

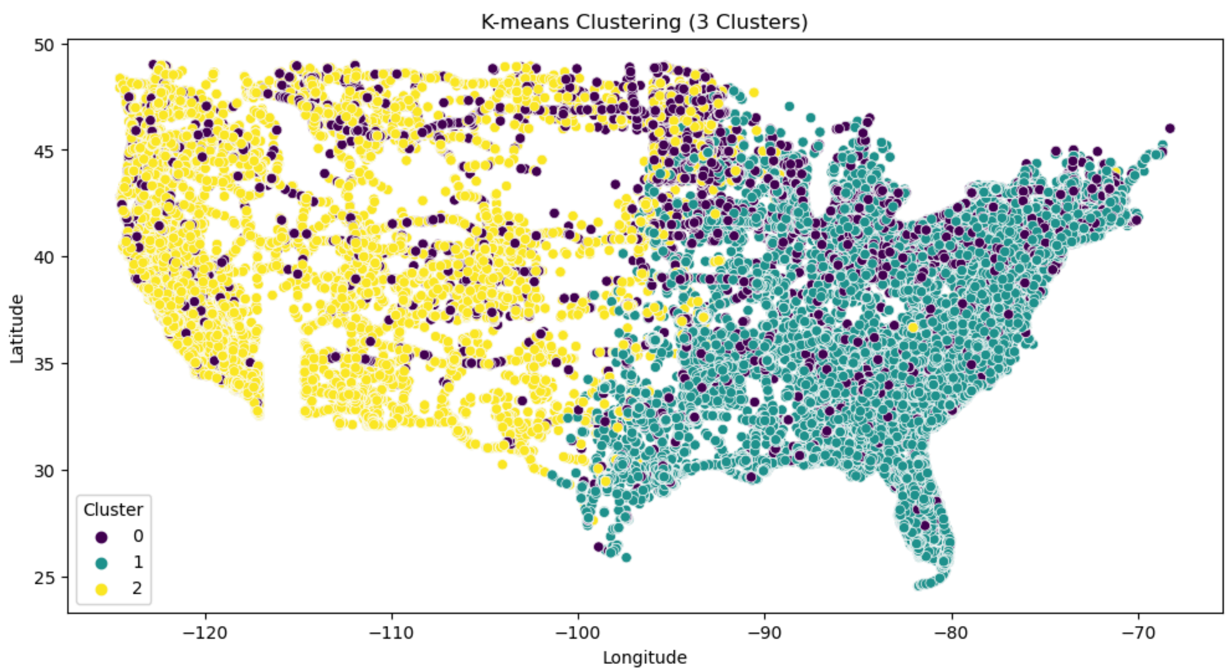
In [34]:
```python
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
import seaborn as sns
```

In [35]:
```python
# Selecting relevant features for clustering (numeric columns)
numeric_features = new_data.select_dtypes(include=['float64', 'int64']).columns
new_data_numeric = new_data[numeric_features]
```

In [36]:
```python
# Standardizing the numeric features
scaler = StandardScaler()
new_data_scaled = scaler.fit_transform(new_data_numeric)
```

In [37]:
```python
# Applying K-means clustering with 3 clusters
kmeans = KMeans(n_clusters=3, random_state=42)
new_data['Cluster'] = kmeans.fit_predict(new_data_scaled)
```

In [38]:
```python
# Visualizing the clusters
plt.figure(figsize=(12, 6))
sns.scatterplot(x='Start_Lng', y='Start_Lat', hue='Cluster', data=new_data, palette='v
plt.title('K-means Clustering (3 Clusters)')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



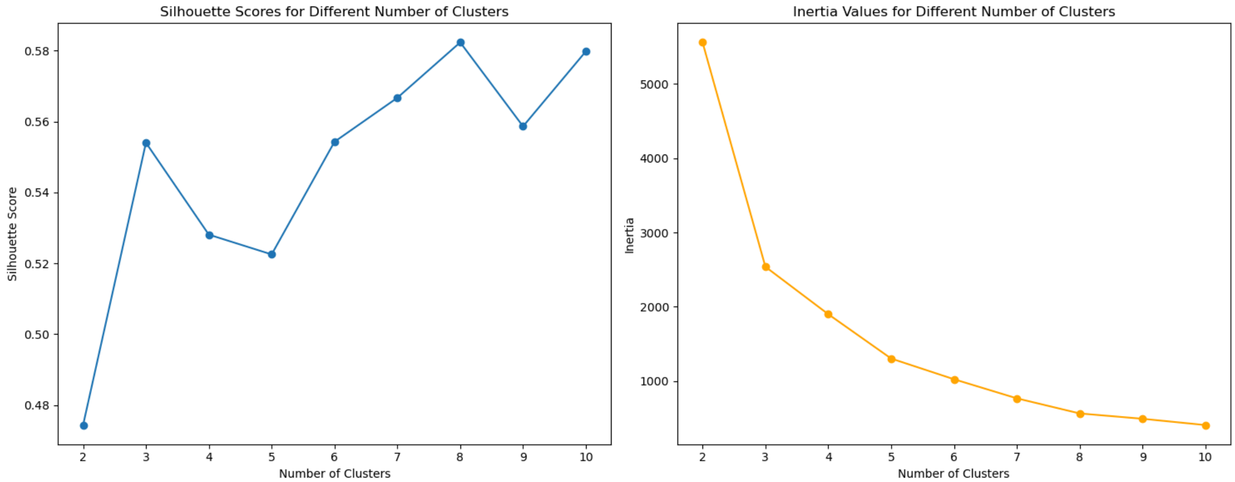
In [39]:
```python
new_data.info()
```

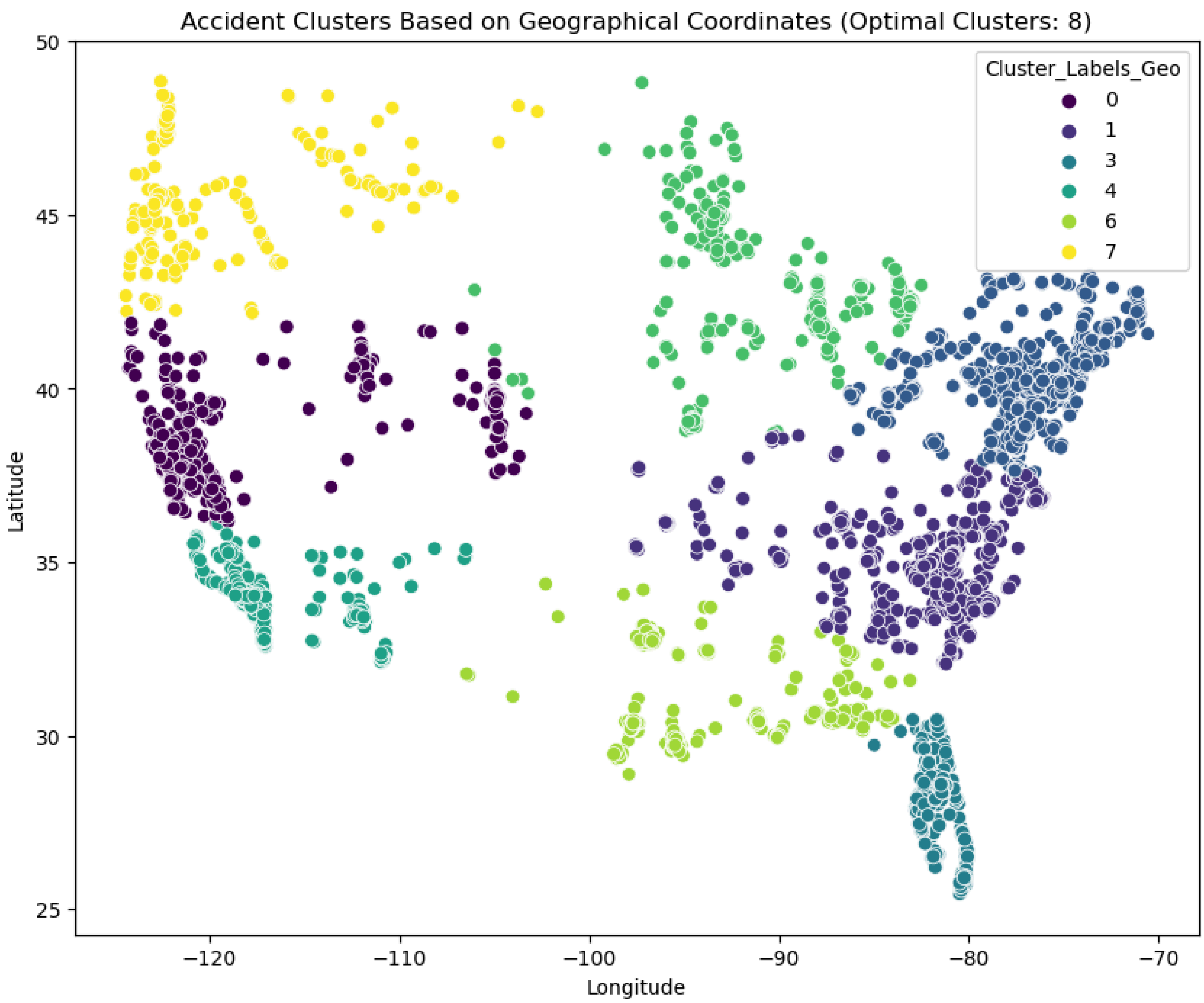
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2009085 entries, 0 to 2009084
Data columns (total 42 columns):
 #   Column                 Dtype  
---  ------                 -----  
 0   ID                     object 
 1   Source                 object 
 2   Severity               int64  
 3   Start_Time             object 
 4   End_Time               object 
 5   Start_Lat              float64
 6   Start_Lng              float64
 7   Distance(mi)           float64
 8   Street                 object 
 9   City                   object 
 10  County                 object 
 11  State                  object 
 12  Zipcode                object 
 13  Country                object 
 14  Timezone               object 
 15  Airport_Code           object 
 16  Weather_Timestamp      object 
 17  Temperature(F)         float64
 18  Humidity(%)            float64
 19  Pressure(in)           float64
 20  Visibility(mi)         float64
 21  Wind_Direction         object 
 22  Wind_Speed(mph)        float64
 23  Weather_Condition      object 
 24  Amenity                bool   
 25  Bump                   bool   
 26  Crossing               bool   
 27  Give_Way               bool   
 28  Junction               bool   
 29  No_Exit                bool   
 30  Railway                bool   
 31  Roundabout             bool   
 32  Station                bool   
 33  Stop                   bool   
 34  Traffic_Calming        bool   
 35  Traffic_Signal         bool   
 36  Turning_Loop           bool   
 37  Sunrise_Sunset         object 
 38  Civil_Twilight         object 
 39  Nautical_Twilight      object 
 40  Astronomical_Twilight  object 
 41  Cluster                int32  
dtypes: bool(13), float64(8), int32(1), int64(1), object(19)
memory usage: 461.8+ MB
```

In [40]:

```
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import silhouette_score

# Creating a smaller DataFrame with 5000 random rows
small_data = new_data.sample(n=5000, random_state=42)

# Extracting relevant columns for geographical clustering
geo_columns = ['Start_Lat', 'Start_Lng']
geo_data = small_data[geo_columns]
```

```
# Standardizing the data
scaler = StandardScaler()
geo_data_scaled = scaler.fit_transform(geo_data)

# Determining the optimal number of clusters using silhouette score and inertia
silhouette_scores_geo = []
inertia_values_geo = []

for n_clusters in range(2, 11):
    kmeans_geo = KMeans(n_clusters=n_clusters, random_state=42)
    cluster_labels_geo = kmeans_geo.fit_predict(geo_data_scaled)

    silhouette_avg_geo = silhouette_score(geo_data_scaled, cluster_labels_geo)
    silhouette_scores_geo.append(silhouette_avg_geo)

    inertia_value_geo = kmeans_geo.inertia_
    inertia_values_geo.append(inertia_value_geo)

# Finding the optimal number of clusters based on silhouette score
optimal_clusters_geo_silhouette = silhouette_scores_geo.index(max(silhouette_scores_ge

# Find the optimal number of clusters based on inertia
optimal_clusters_geo_inertia = inertia_values_geo.index(min(inertia_values_geo)) + 2

# Plotting silhouette scores
plt.figure(figsize=(15, 6))

plt.subplot(1, 2, 1)
plt.plot(range(2, 11), silhouette_scores_geo, marker='o')
plt.title('Silhouette Scores for Different Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Silhouette Score')

# Plotting inertia values
plt.subplot(1, 2, 2)
plt.plot(range(2, 11), inertia_values_geo, marker='o', color='orange')
plt.title('Inertia Values for Different Number of Clusters')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')

plt.tight_layout()
plt.show()

# Performing K-means clustering with the optimal number of clusters (We can choose eit
kmeans_geo_optimal = KMeans(n_clusters=optimal_clusters_geo_silhouette, random_state=4
cluster_labels_geo_optimal = kmeans_geo_optimal.fit_predict(geo_data_scaled)

# Adding cluster labels to the original dataframe
small_data['Cluster_Labels_Geo'] = cluster_labels_geo_optimal

# Visualizing clusters on a scatter plot
plt.figure(figsize=(10, 8))
sns.scatterplot(x='Start_Lng', y='Start_Lat', hue='Cluster_Labels_Geo', data=small_dat
plt.title(f'Accident Clusters Based on Geographical Coordinates (Optimal Clusters: {op
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```

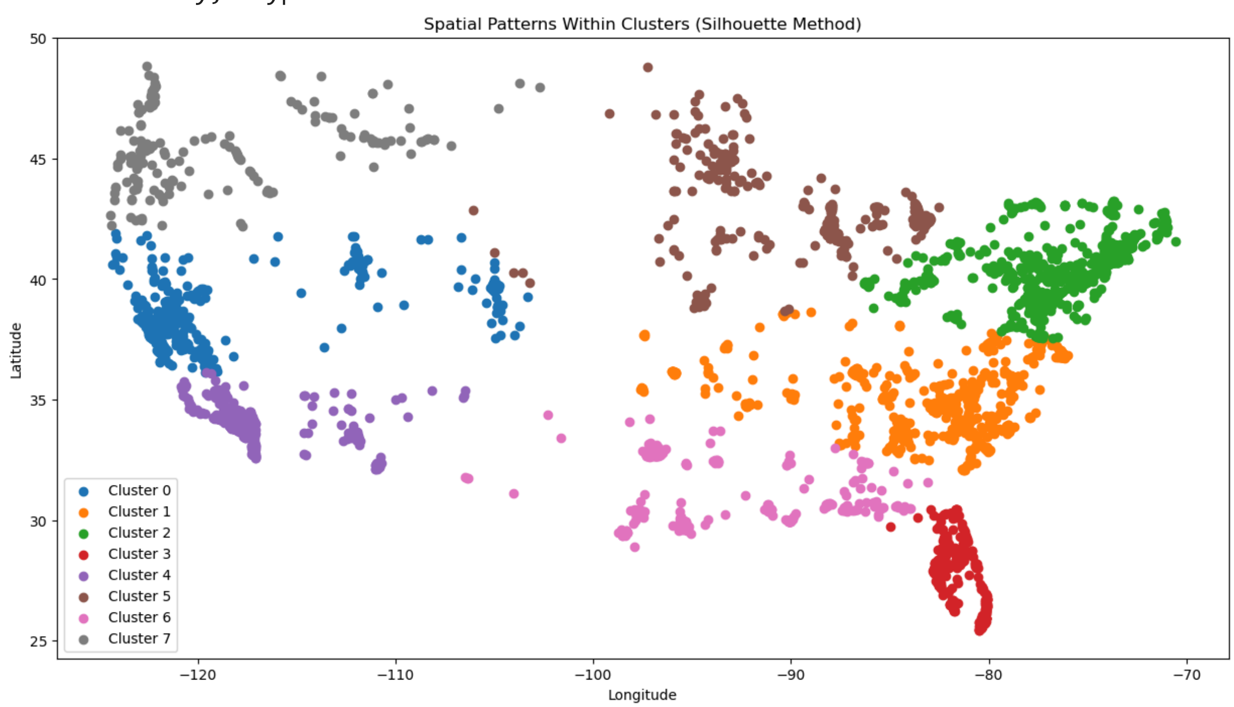


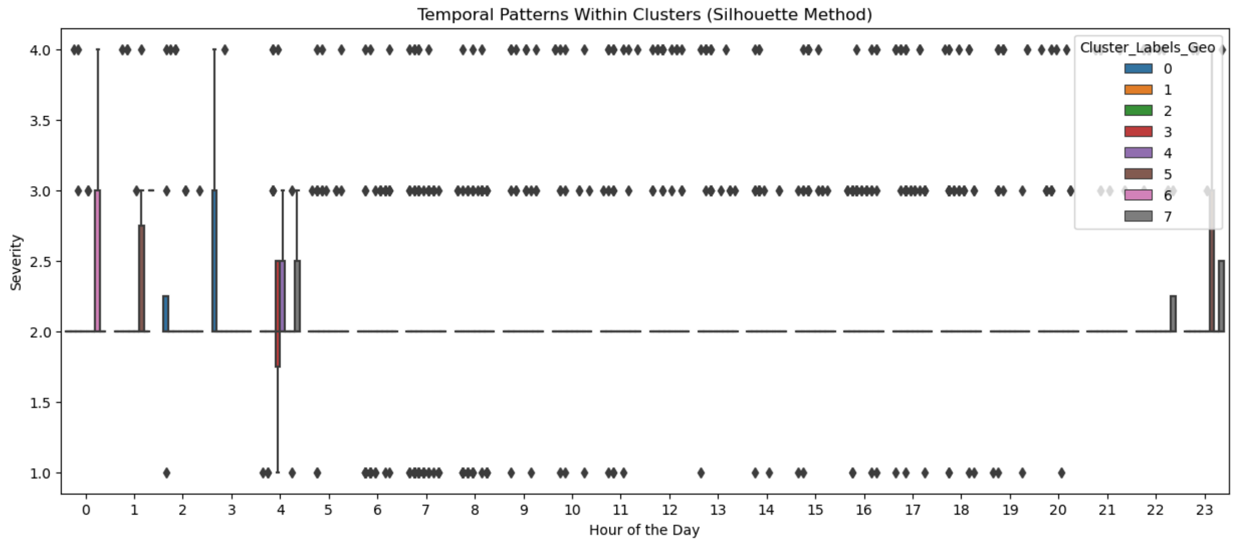


In [41]:
```
severity_column = 'Severity'

# 1. Cluster Statistics
cluster_statistics = small_data.groupby('Cluster_Labels_Geo')[severity_column].mean()
print("Cluster Statistics - Average Severity:")
print(cluster_statistics)

# 2. Spatial Patterns
plt.figure(figsize=(15, 8))
for cluster_label in range(optimal_clusters_geo_silhouette):
    cluster_data = small_data[small_data['Cluster_Labels_Geo'] == cluster_label]
    plt.scatter(cluster_data['Start_Lng'], cluster_data['Start_Lat'], label=f'Cluster
```

```
plt.title('Spatial Patterns Within Clusters (Silhouette Method)')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.legend()
plt.show()

# 3. Temporal Patterns
if 'Start_Time' in small_data.columns:
    small_data['Start_Time'] = pd.to_datetime(small_data['Start_Time'])
    small_data['Hour'] = small_data['Start_Time'].dt.hour

    plt.figure(figsize=(15, 6))
    sns.boxplot(x='Hour', y=severity_column, hue='Cluster_Labels_Geo', data=small_data
    plt.title('Temporal Patterns Within Clusters (Silhouette Method)')
    plt.xlabel('Hour of the Day')
    plt.ylabel('Severity')
    plt.show()
```

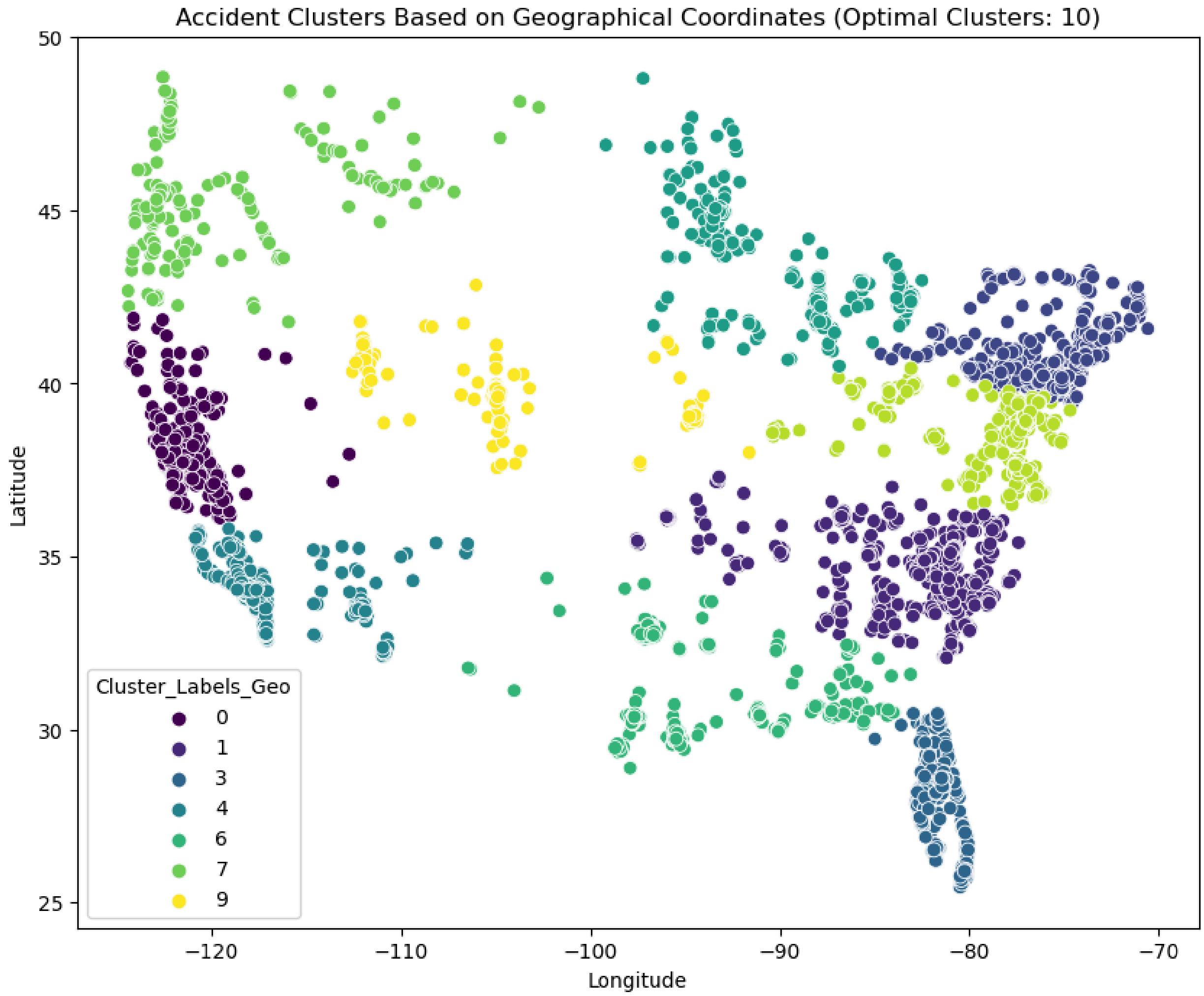
```
Cluster Statistics - Average Severity:
Cluster_Labels_Geo
0    2.043333
1    2.070696
2    2.121884
3    2.015267
4    2.051862
5    2.110837
6    2.076389
7    2.076923
Name: Severity, dtype: float64
```





In [42]:

```
kmeans_geo_optimal = KMeans(n_clusters=optimal_clusters_geo_inertia , random_state=42)
cluster_labels_geo_optimal = kmeans_geo_optimal.fit_predict(geo_data_scaled)

# Adding cluster labels to the original dataframe
small_data['Cluster_Labels_Geo'] = cluster_labels_geo_optimal

# Visualizing clusters on a scatter plot
plt.figure(figsize=(10, 8))
sns.scatterplot(x='Start_Lng', y='Start_Lat', hue='Cluster_Labels_Geo', data=small_dat
plt.title(f'Accident Clusters Based on Geographical Coordinates (Optimal Clusters: {op
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```

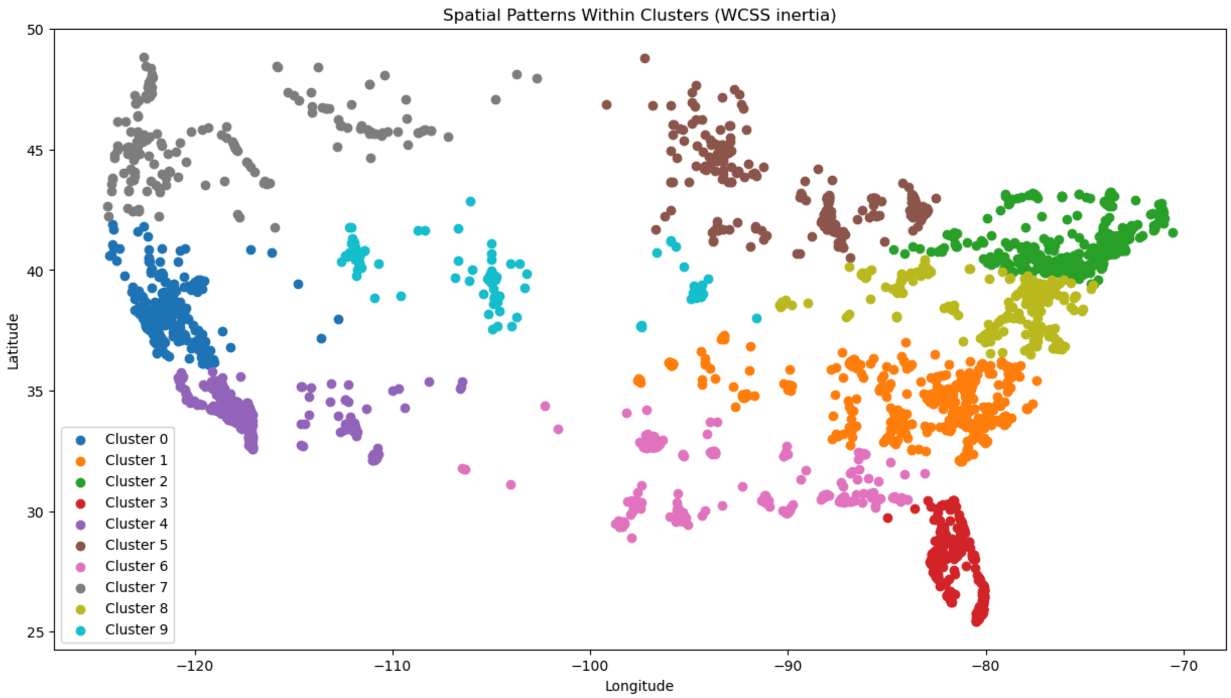


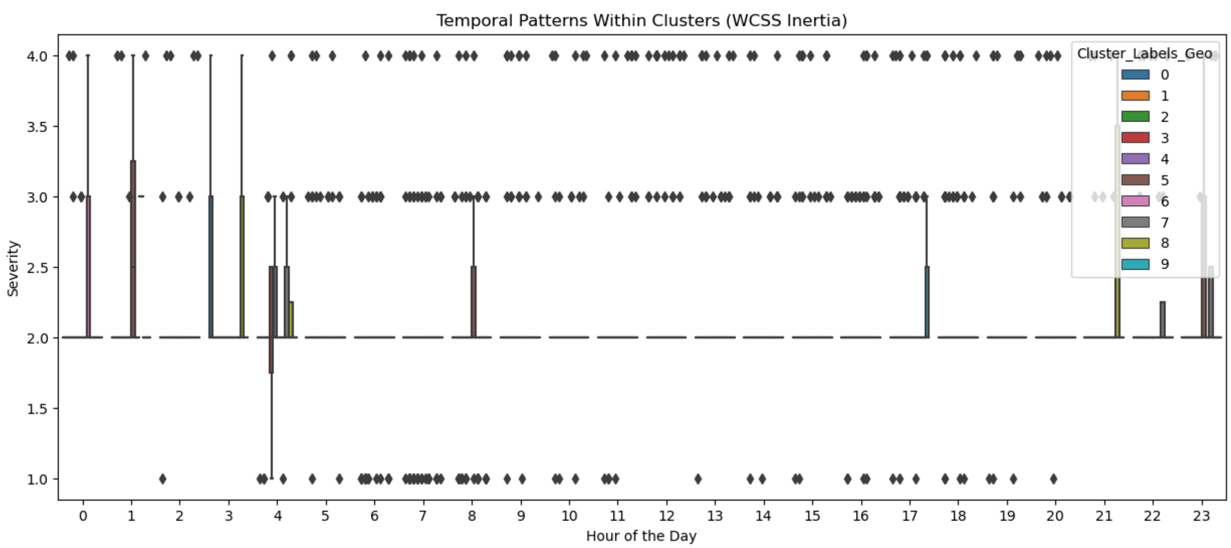
In [43]:

```
severity_column = 'Severity'

# 1. Cluster Statistics
cluster_statistics = small_data.groupby('Cluster_Labels_Geo')[severity_column].mean()
print("Cluster Statistics - Average Severity:")
print(cluster_statistics)

# 2. Spatial Patterns
plt.figure(figsize=(15, 8))
for cluster_label in range(optimal_clusters_geo_inertia):
    cluster_data = small_data[small_data['Cluster_Labels_Geo'] == cluster_label]
    plt.scatter(cluster_data['Start_Lng'], cluster_data['Start_Lat'], label=f'Cluster

plt.title('Spatial Patterns Within Clusters (WCSS inertia)')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.legend()
plt.show()

# 3. Temporal Patterns
if 'Start_Time' in small_data.columns:
    small_data['Start_Time'] = pd.to_datetime(small_data['Start_Time'])
    small_data['Hour'] = small_data['Start_Time'].dt.hour

    plt.figure(figsize=(15, 6))
    sns.boxplot(x='Hour', y=severity_column, hue='Cluster_Labels_Geo', data=small_data
    plt.title('Temporal Patterns Within Clusters (WCSS Inertia)')
```

```
    plt.xlabel('Hour of the Day')
    plt.ylabel('Severity')
    plt.show()
```

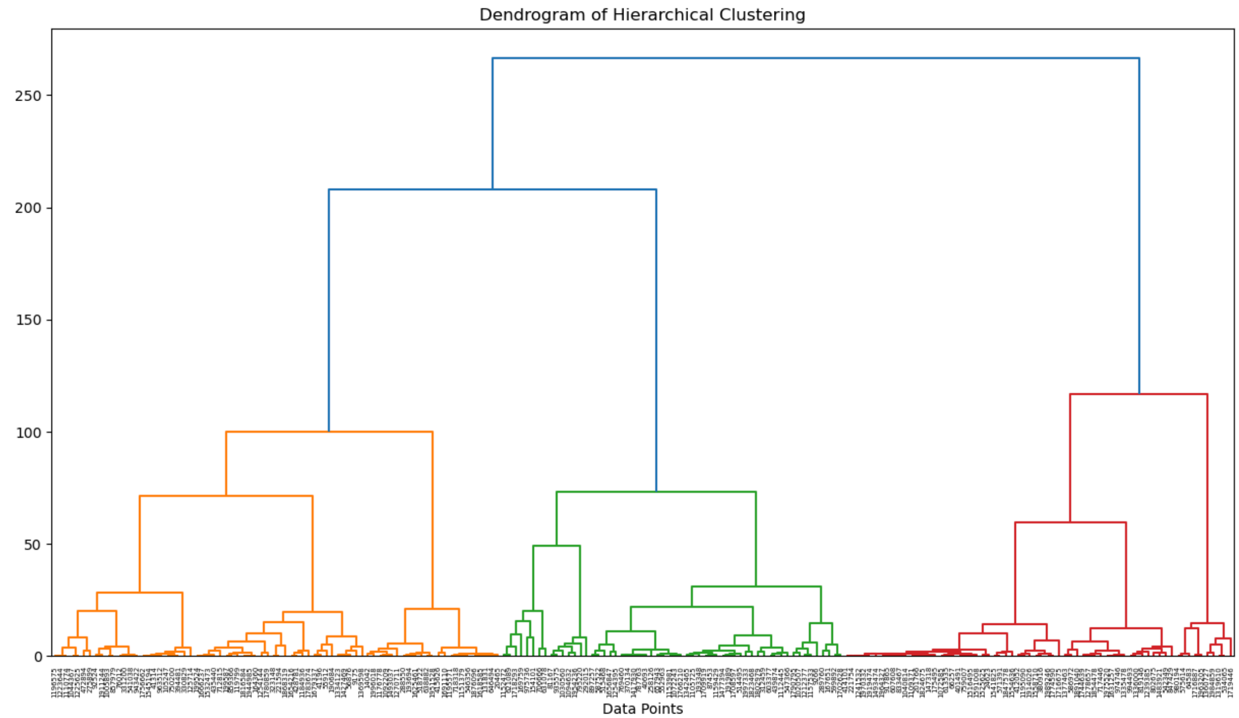
```
Cluster Statistics - Average Severity:
Cluster_Labels_Geo
0    2.026423
1    2.066757
2    2.093496
3    2.015267
4    2.052000
5    2.101744
6    2.076744
7    2.076531
8    2.155102
9    2.134503
Name: Severity, dtype: float64
```

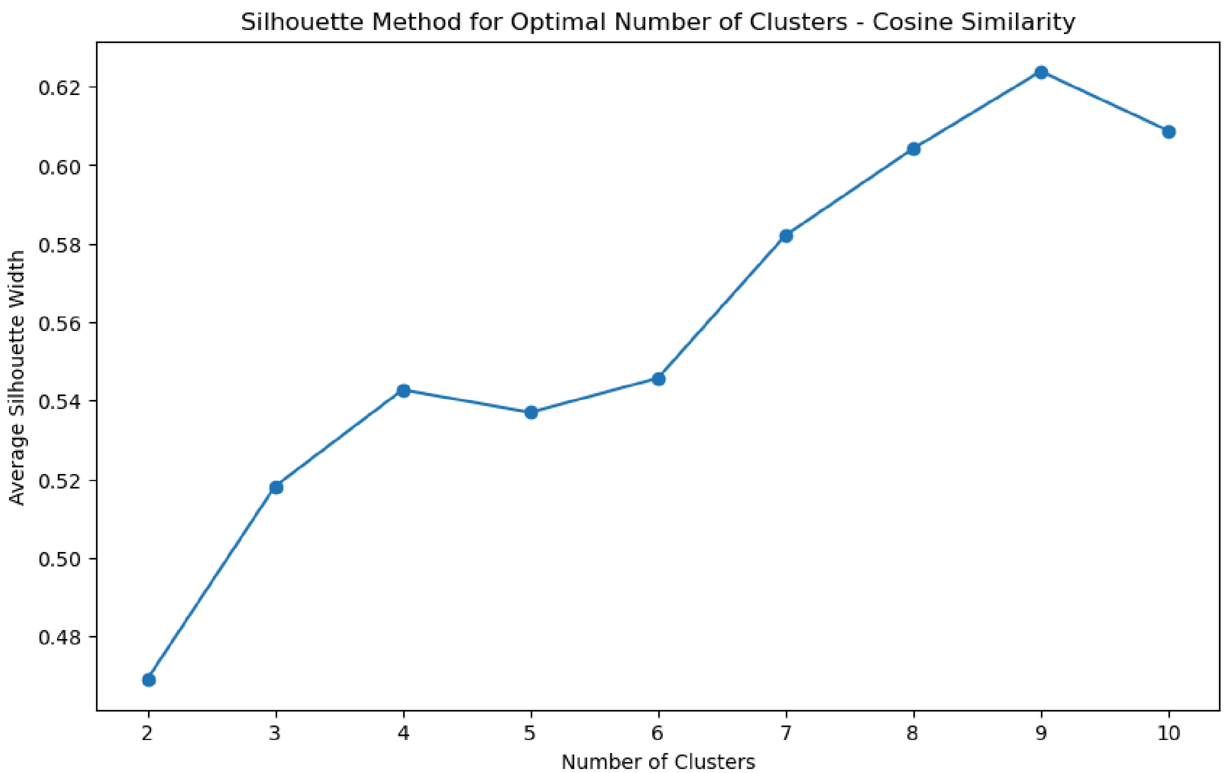


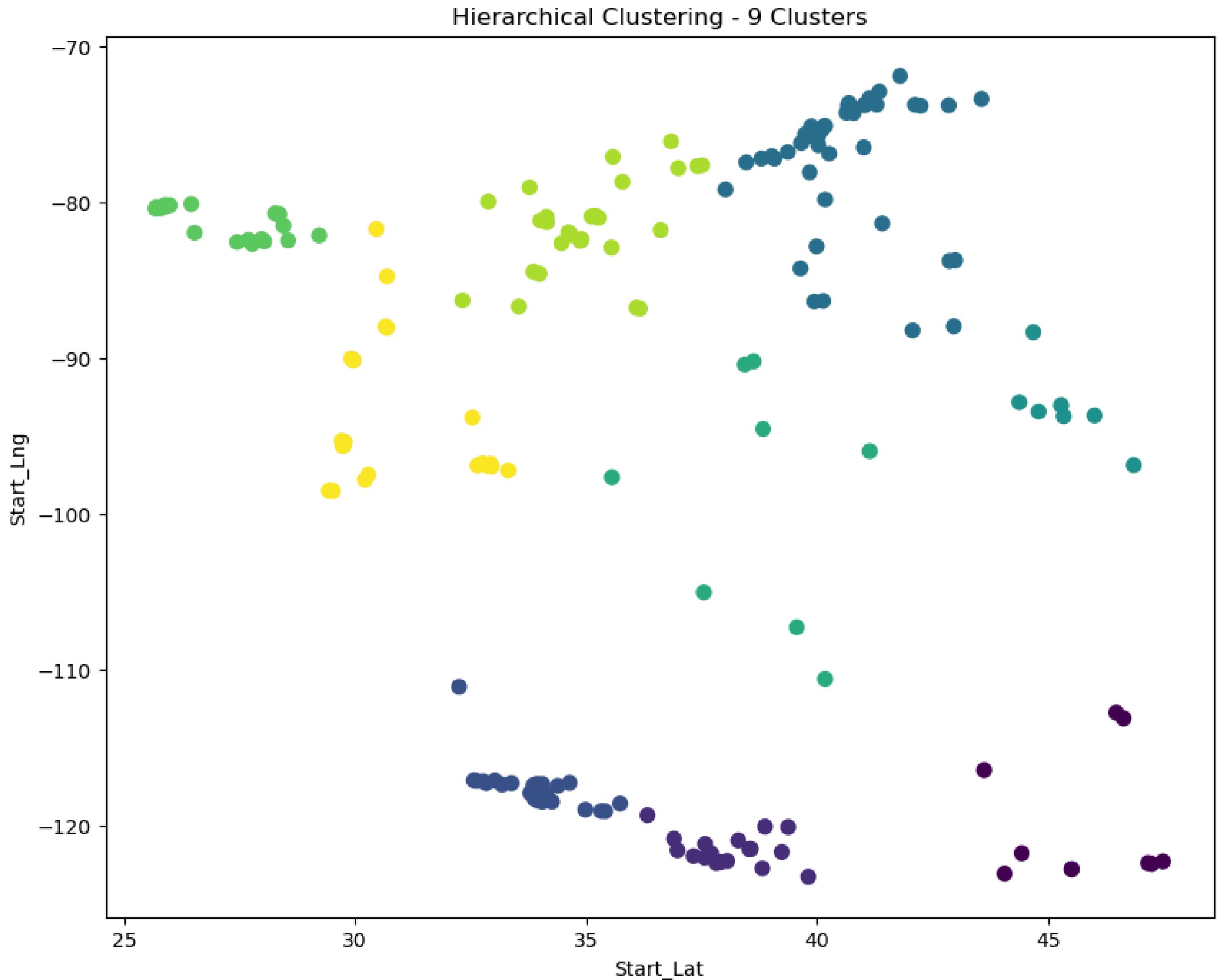


In [49]:
```python
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.metrics import silhouette_score
from scipy.cluster.hierarchy import linkage, dendrogram, fcluster
import matplotlib.pyplot as plt

selected_columns = ['Start_Lat', 'Start_Lng']

sample_size = 200
new_data_sample = new_data.sample(n=sample_size)

# Extracting the selected columns for clustering
X = new_data_sample[selected_columns]

scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

cosine_dist = 1 - np.dot(X_scaled, X_scaled.T)

# Performing hierarchical clustering with ward linkage
linkage_matrix = linkage(cosine_dist, method='ward')

# Plotting the dendrogram
plt.figure(figsize=(15, 8))
dendrogram(linkage_matrix, labels=new_data_sample.index.values, orientation='top', dis
plt.title('Dendrogram of Hierarchical Clustering')
plt.xlabel('Data Points')
plt.show()

# Determining the optimal number of clusters based on the highest silhouette score
silhouette_scores = []
for k in range(2, 11):
    cluster_labels = fcluster(linkage_matrix, k, criterion='maxclust')
    silhouette_avg = silhouette_score(cosine_dist, cluster_labels)
    silhouette_scores.append(silhouette_avg)

optimal_clusters = np.argmax(silhouette_scores) + 2

# Determining clusters
clusters = fcluster(linkage_matrix, optimal_clusters, criterion='maxclust')

# Plotting silhouette scores using a line plot
plt.figure(figsize=(10, 6))
plt.plot(range(2, 11), silhouette_scores, marker='o')
plt.title('Silhouette Method for Optimal Number of Clusters - Cosine Similarity')
plt.xlabel('Number of Clusters')
plt.ylabel('Average Silhouette Width')
plt.show()

# Visualizing clusters on a scatter plot
plt.figure(figsize=(10, 8))
plt.scatter(X.iloc[:, 0], X.iloc[:, 1], c=clusters, cmap='viridis', s=50)
plt.title(f'Hierarchical Clustering - {optimal_clusters} Clusters')
plt.xlabel('Start_Lat')
plt.ylabel('Start_Lng')
plt.show()
```


Dendrogram of Hierarchical Clustering
250
200
150
100
50
0
Data Points


Silhouette Method for Optimal Number of Clusters - Cosine Similarity
Average Silhouette Width
0.62
0.60
0.58
0.56
0.54
0.52
0.50
0.48
2
3
4
5
6
7
8
9
10
Number of Clusters


Hierarchical Clustering - 9 Clusters
Start_Lng
−70
−80
−90
−100
−110
−120
Start_Lat
25
30
35
40
45

In [ ]: