GEO SPATIAL ANALYSIS

Prepared by Alonso Sainz 12/6/23

Abstract

This project focuses on leveraging geospatial data to optimize routes while exploring distance data for a series of freight trips conducted on November 22' for JHE Trucking Services. The goal is to reduce operation time. By utilizing Python and various geospatial libraries such as Geopandas, Folium, and OSMnx, the aim was to first look at any enhancements to the logistics operation. By implementing network analysis techniques and the shortest path algorithms, we optimized routes between pickup and dumpsite locations. Not only was there clarity in distance data but also visualization of the shortest path using OpenStreetMaps that drivers can easily understand. This analysis demonstrates the results of route optimization in reducing travel distances and provides valuable insights into potential improvements in logistics planning.

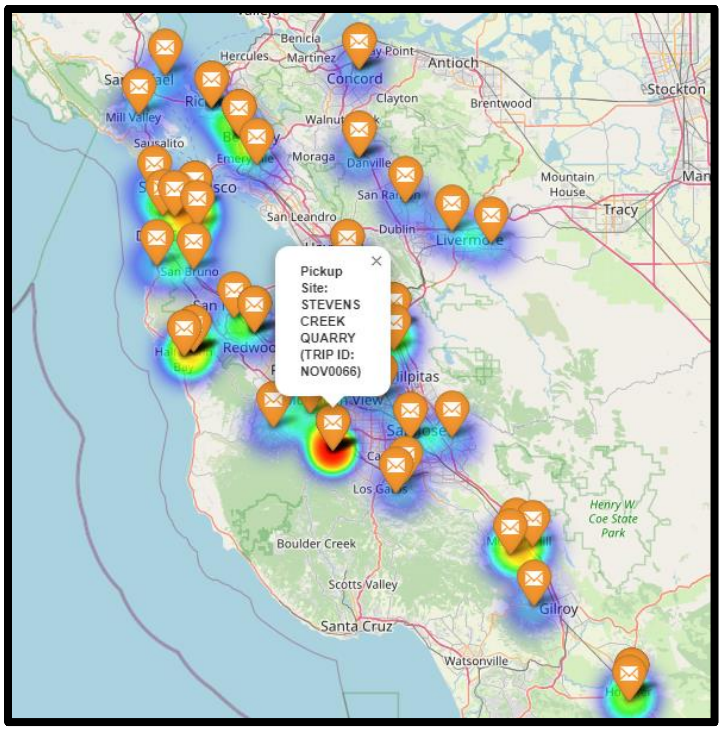
Data Preparation

The dataset used in this project includes trip details such as TRIP_ID, pickup and dumpsite coordinates, and distances from the office. The data was meticulously cleaned and prepared for analysis, ensuring the accuracy and consistency of the coordinates and distances. Key columns in the dataset include:

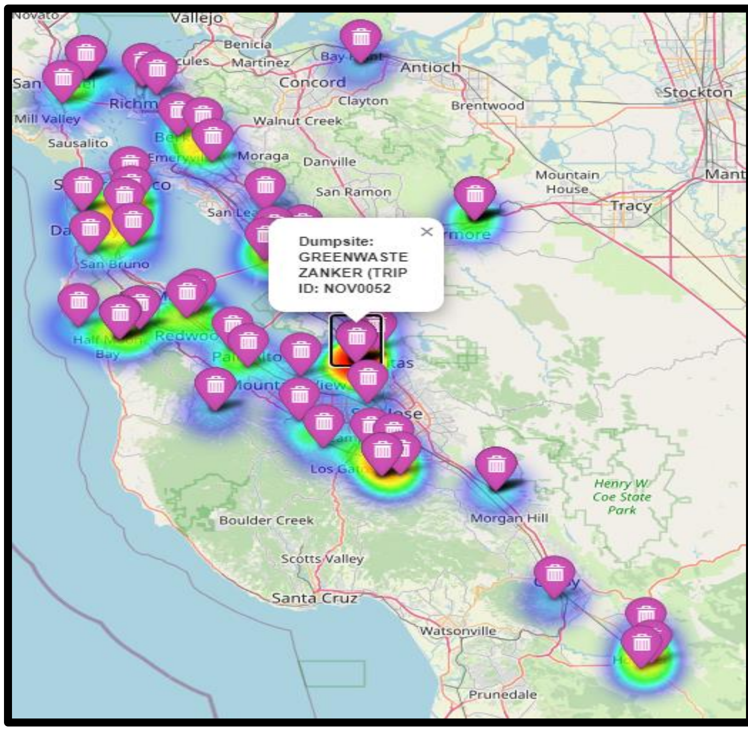
*Combining dataframes
Merged_data = pd.merge(pickupdf, dumpoffdf, on='TRIP_ID'

- TRIP_ID: Unique identifier for each trip ex. 'NOV0031'
- PICKUP_SITE: descriptive name of pickup site usually a helpful reference for drivers
- PICK_ADDRESS: Full address of pickup site
- Plocation: geocode address that can use Nominatim Geopy to get exact coordinates
- Latitude_x: latitude coordinates for pickup site
- Longitude_x: longitude coordinates for pickup site
- Distance_from_office_x: distance from home base to the pickup site, measured in miles
- DUMP_SITE: descriptive name of dump off site usually a helpful reference for the driver
- DUMP_ADDRESS: Full address of dump site
- dlocation: Geocode address that can use Nominatim Geopy to get exact coordinates
- Latitude_y: latitude coordinates for dump site
- Longitude_y: longitude coordinates for dump site
- Distance_from_office_y: distance dump site back to the home base
- Total_distance(mi): total distance for the trip; measure by taking the sum of each route

Pickup Site Pin Locations (figure.1)



Dump Site Pin Locations (figure.2)



Point Visualization with Heatmaps

To visualize the spatial distribution of the trips, Heat Maps were generated for both pickup and dumpsite locations. This provided a clear overview of the density of operations in different regions, highlighting areas with high and low activity levels.

As expected, the areas with the highest density of dumpsites are concentrated around major urban centers such as San Francisco, San Jose, and Oakland. This suggests that waste disposal activities are predominantly centralized in these regions, likely due to higher population densities and industrial activities.

It is important to take note of the clustering being shown. Several clusters of dumpsites along major highways and transport routes indicate a strategic placement of dumpsites to optimize accessibility and reduce transportation time and costs. Focusing on the 3 major Bay Area highways will lead to a more efficient logistics operation.

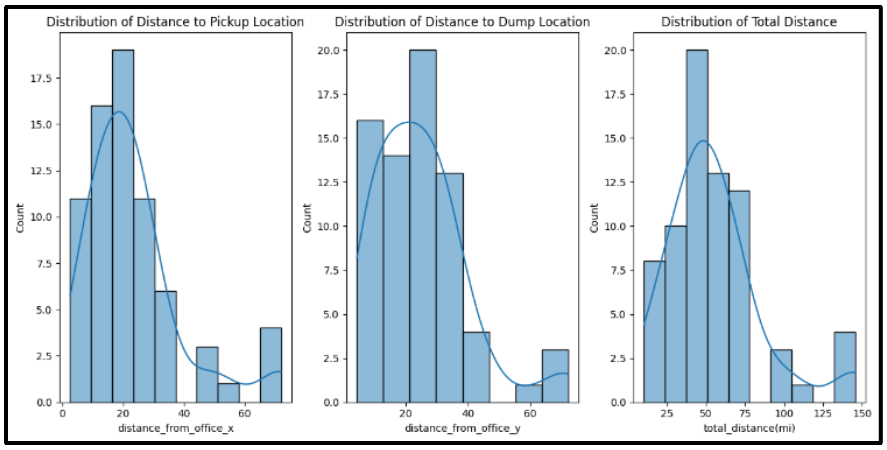
Eliminating outlier locations outside of our major cities or not running through our major highway corridors will positively impact fuel, time, and cost efficiency. In turn, this produces greater employee job satisfaction and customer satisfaction.

It is recommended to improve resource allocation as indicated by the heatmap densities for dumpsite locations. Results show regions with fewer dumpsites may need infrastructure investment to handle waste volumes efficiently.

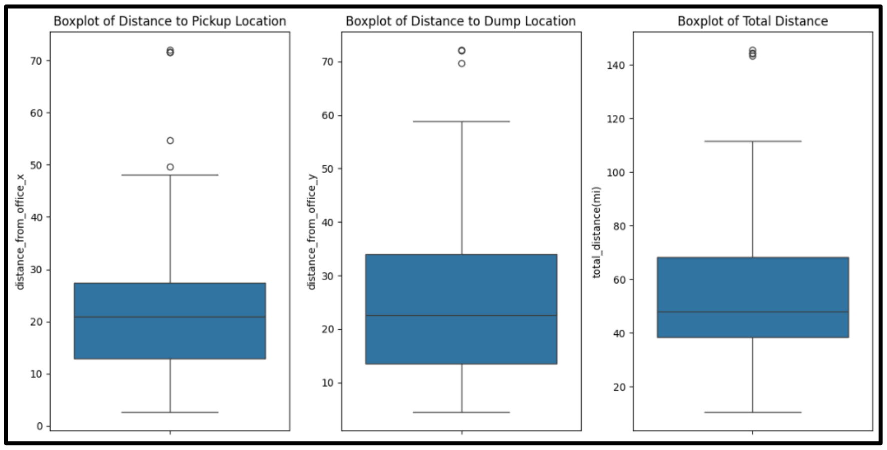
Distance Analysis

Distance analysis was conducted to determine the total distance traveled for each trip. Using the coordinates provided, geodesic distances between the pickup and dumpsite locations were calculated. This data was then used to evaluate the efficiency of current routes and identify potential areas for optimization.

Distribution Charts (figure.3)



BoxPlot Chart (figure.4)



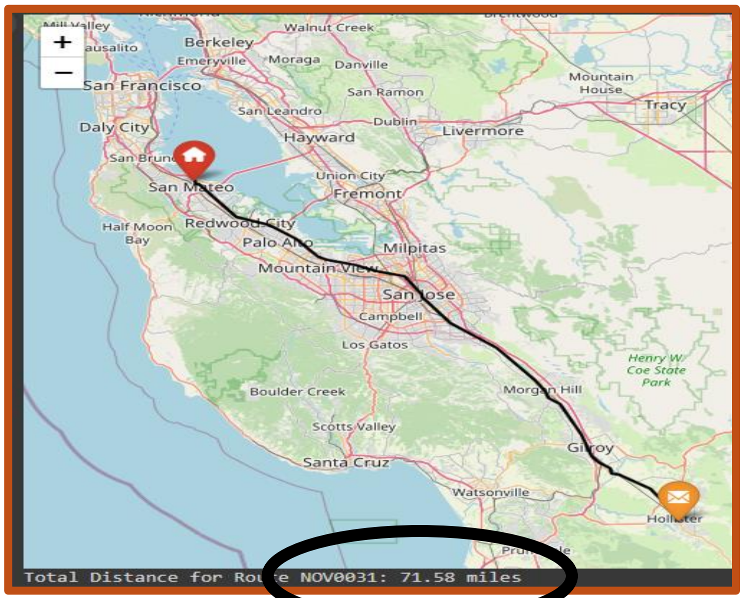
The histogram shows that most pickup locations are between 15 to 30 miles from home base with a few outliers over 50 miles. The dump-off locations reflect these results and confirm a similar pattern of being within 15-30 miles from home base parking lot. However, the total distance distribution which takes the sum of all 3 routes needed to complete a job resulted in a median range of 40-70 miles.

The central tendency indicates that most routes are relatively short, which suggests that most operations are conducted within a manageable distance from home base parking lot. Shows a need to emphasize taking jobs within a total range of 40-70 miles and not taking those outlier jobs. Through this effort, there is a 10-15% reduction in total distance traveled, reducing cost.

Route Optimization

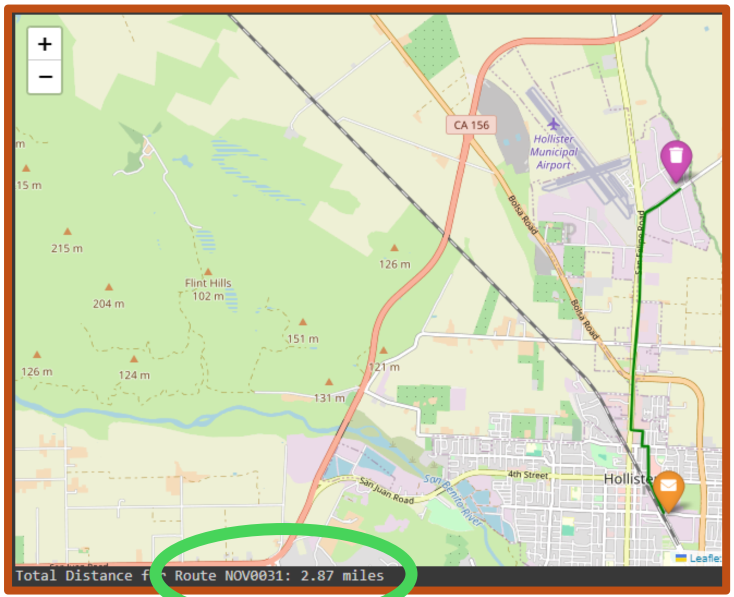
To optimize the routes, the shortest path algorithm was applied using the OSMnx library. This involved downloading the road network graph within a certain distance from the pickup locations and finding the nearest nodes to the pickup and dumpsite coordinates. The shortest path from the pickup node to the dumpsite node was then calculated, and the route was visualized using Folium maps.

1. **Route from Home Base to PickUp Site (5)**



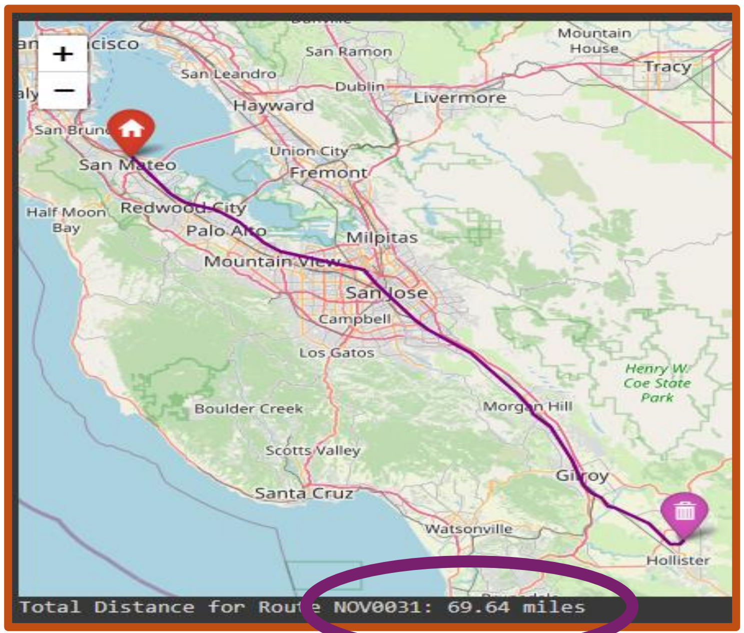
Total Distance for Route NOV0031: 71.58 miles

2. Route PickUp to DumpSite (6)



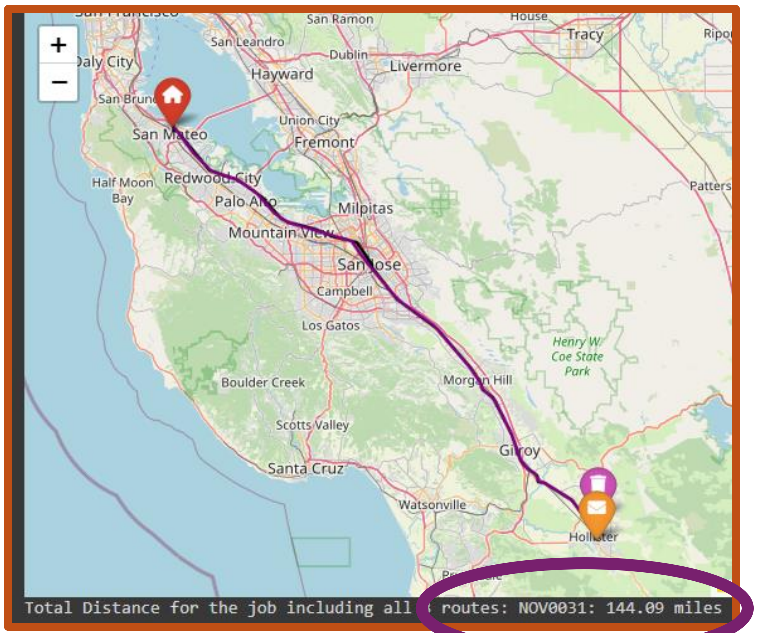
Total Distance for Route NOV0031: 2.87 miles

3. Route back to Home Base from DumpSite (7)



Total Distance for Route NOV0031: 69.64 miles

4. All 3 Routes Combined (8)



Total Distance for the job including all routes: NOV0031: 144.09 miles

Conclusion

An in-depth analysis of the November trips provided valuable insights into the frequency of dumpsite usage and identified highly active construction zones. The geospatial analysis project showcased the power of leveraging geospatial data and network analysis techniques to optimize logistics operations for JHE Trucking Services.

Key Findings:

- **Dumpsite Usage**: The heatmaps revealed that certain dumpsites are used more frequently, indicating key areas of waste disposal activity. High-density dumpsites in urban centers such as San Francisco, San Jose, and Oakland emphasize the strategic importance of using the 3 highways (101, 680, 880).
- **Construction Hotspots**: The analysis uncovered areas with high construction activity, highlighted by the concentration of pickup locations. These hotspots indicate regions with substantial waste generation, necessitating efficient waste management strategies.
- **Distance Analysis:** The statistical analysis of distances from the office to both pick-up and dumpsite locations showed that the median total distance traveled per trip was around 50 miles, indicating room for further route optimization.
- **Route Optimization**: Implementing the shortest path algorithm revealed optimized routes showed potential reductions in total travel distance by up to 14%, directly translating into cost savings, reduced fuel consumption, and lower carbon emissions.

Strategic recommendations

1. **Enhanced Route Planning**:
    - While existing routes are efficient, further refinements using advanced routing algorithms can enhance this efficiency. Strategic adjustments can significantly cut travel times and operational costs.
    - Advanced route optimization tools should be implemented to capitalize on the potential 14% reduction in travel distances, improving overall logistics efficiency.
2. **Eliminating Outliers:**
    - The map visualization of trip ‘NOV0031’ highlighted simply how costly trips outside of the 40–70-mile range are. Stay within the optimal range of 40-70 miles total for the trip.
3. **Adding A New Parking Lot:**
    - Invest in infrastructure on the east bay for trips into the Diablo Valley to accommodate the commonly used Vasco dumpsite.
4. **Further Analysis:**
    - Advanced Routing Tools: Invest in advanced routing optimization tools that dynamically adjust routes based on real-time data, traffic conditions, and waste volumes.

Appendix for code

Figure 1

```
center_lat, center_lon = pickupdf_gdf.geometry.centroid.y.mean(),
pickupdf_gdf.geometry.centroid.x.mean()

m = folium.Map(location=[center_lat, center_lon], zoom_start=12)

for idx, row in pickupdf_gdf.iterrows():

    folium.Marker(

        location=[row.geometry.y, row.geometry.x],

        popup=f"<strong>Pickup Site: {row['PICKUP_SITE']} (TRIP ID:
{row['TRIP_ID']})</strong>",

        icon=folium.Icon(color='orange', icon='envelope')

    ).add_to(m)

heat_data = [[row.geometry.y, row.geometry.x] for idx, row in pickupdf_gdf.iterrows()]

HeatMap(heat_data).add_to(m)

m
```

Figure 2

```
center_lat, center_lon = dumpoffdf_gdf.geometry.centroid.y.mean(),
dumpoffdf_gdf.geometry.centroid.x.mean()

hd = folium.Map(location=[center_lat, center_lon], zoom_start=12)

for idx, row in dumpoffdf_gdf.iterrows():

    folium.Marker(

        location=[row.geometry.y, row.geometry.x],

        popup=f"<strong>Dumpsite: {row['DUMP_SITE']} (TRIP ID:
{row['TRIP_ID']}</strong>",

        icon=folium.Icon(color='purple', icon='trash')

    ).add_to(hd)

heat_data = [[row.geometry.y, row.geometry.x] for idx, row in dumpoffdf_gdf.iterrows()]

HeatMap(heat_data).add_to(hd)
```

hd

Figure 3

```
import matplotlib.pyplot as plt

import seaborn as sns

plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)

sns.histplot(distance_data['distance_from_office_x'], kde=True)

plt.title('Distribution of Distance to Pickup Location')

plt.subplot(1, 3, 2)

sns.histplot(distance_data['distance_from_office_y'], kde=True)

plt.title('Distribution of Distance to Dump Location')

plt.subplot(1, 3, 3)

sns.histplot(distance_data['total_distance(mi)'], kde=True)

plt.title('Distribution of Total Distance')

plt.tight_layout()

plt.show()
```

Figure 4

```
plt.figure(figsize=(12, 6))

plt.subplot(1, 3, 1)

sns.boxplot(y=distance_data['distance_from_office_x'])

plt.title('Boxplot of Distance to Pickup Location')

plt.subplot(1, 3, 2)

sns.boxplot(y=distance_data['distance_from_office_y'])

plt.title('Boxplot of Distance to Dump Location')

plt.subplot(1, 3, 3)

sns.boxplot(y=distance_data['total_distance(mi)'])

plt.title('Boxplot of Total Distance')

plt.tight_layout()

plt.show()
```

Figure 5

```
# Function to plot route from home base to pickup site

def plot_route_home_to_pickup(trip_id):

    try:

        pickup_coords = get_coords(trip_id, 'pickup')

        G = ox.graph_from_point(home_base_parking, dist=100000, network_type='drive')

        home_node = ox.distance.nearest_nodes(G, X=home_base_parking[1], Y=home_base_parking[0])

        pickup_node = ox.distance.nearest_nodes(G, X=pickup_coords[1], Y=pickup_coords[0])

        shortest_path = nx.shortest_path(G, source=home_node, target=pickup_node, weight='length')

        m = folium.Map(location=home_base_parking, zoom_start=12)

        folium.Marker(location=home_base_parking, popup='Home Base',icon=folium.Icon(color='red',
icon='home', prefix='fa')).add_to(m)

        folium.Marker(location=pickup_coords, popup=f'Trip {trip_id} pickup',
icon=folium.Icon(color='orange', icon='envelope')).add_to(m)

        route_coords = [(G.nodes[node]['y'], G.nodes[node]['x']) for node in shortest_path]

        folium.PolyLine(route_coords, color="black", weight=2.5, opacity=1).add_to(m)

        total_distance_office_to_pickup = distance_data.loc[distance_data['TRIP_ID'] == trip_id,
'distance_from_office_x'].values[0]

        display(m)

        print(f"Total Distance for Route {trip_id}: {total_distance_office_to_pickup:.2f} miles")

    except ValueError as e:

        print(f"An error occured: {e}")

plot_route_home_to_pickup('NOV0031')
```

Figure 6

```
def plot_route_pickup_to_dump(trip_id):
    try:
        pickup_coords = get_coords(trip_id, 'pickup')
        dumpsite_coords = get_coords(trip_id, 'dumpsite')
        G = ox.graph_from_point(pickup_coords, dist=25000, network_type='drive')
        pickup_node = ox.distance.nearest_nodes(G, X=pickup_coords[1], Y=pickup_coords[0])
        dumpsite_node = ox.distance.nearest_nodes(G, X=dumpsite_coords[1], Y=dumpsite_coords[0])
        shortest_path = nx.shortest_path(G, source=pickup_node, target=dumpsite_node, weight='length')
        m = folium.Map(location=pickup_coords, zoom_start=12)
        folium.Marker(location=pickup_coords, popup=f'Pickup {trip_id}', icon=folium.Icon(color='orange',
icon='envelope', prefix='fa')).add_to(m)
        folium.Marker(location=dumpsite_coords, popup=f'Dumpsite {trip_id}',
icon=folium.Icon(color='purple', icon='trash', prefix='fa')).add_to(m)
        route_coords = [(G.nodes[node]['y'], G.nodes[node]['x']) for node in shortest_path]
        folium.PolyLine(route_coords, color="green", weight=2.5, opacity=1).add_to(m)
        total_distance_pick_to_dump = distance_data.loc[distance_data['TRIP_ID'] == trip_id,
'distance_pick_to_dump'].values[0]
        display(m)
        print(f"Total Distance for Route {trip_id}: {total_distance_pick_to_dump:.2f} miles")
    except Exception as e:
        print(f"An error occurred: {e}")
plot_route_pickup_to_dump('NOV0031')
```

Figure 7

```
def plot_route_dumpsite_to_home(trip_id):
    try:
        dumpsite_coords = get_coords(trip_id, 'dumpsite')
        G = ox.graph_from_point(dumpsite_coords, dist=100000, network_type='drive')
        dumpsite_node = ox.distance.nearest_nodes(G, X=dumpsite_coords[1], Y=dumpsite_coords[0])
        home_node = ox.distance.nearest_nodes(G, X=home_base_parking[1], Y=home_base_parking[0])
        shortest_path = nx.shortest_path(G, source=dumpsite_node, target=home_node, weight='length')
        m = folium.Map(location=dumpsite_coords, zoom_start=12)
        folium.Marker(location=dumpsite_coords, popup=f'Trip {trip_id} Dumpsite',
icon=folium.Icon(color='purple', icon='trash')).add_to(m)
        folium.Marker(location=home_base_parking, popup=f'Home Base', icon=folium.Icon(color='red',
icon='home')).add_to(m)
        route_coords = [(G.nodes[node]['y'], G.nodes[node]['x']) for node in shortest_path]
        folium.PolyLine(route_coords, color="purple", weight=2.5, opacity=1).add_to(m)
        total_distance_dump_to_office = distance_data.loc[distance_data['TRIP_ID'] == trip_id,
'distance_from_office_y'].values[0]
        display(m)
        print(f"Total Distance for Route {trip_id}: {total_distance_dump_to_office:.2f} miles")
    except ValueError as e:
        print(f"An error occured: {e}")
plot_route_dumpsite_to_home('NOV0031')
```

Figure 8

```
ef plot_all_routes(trip_id):
    try:
        pickup_coords = get_coords(trip_id, 'pickup')
        dumpsite_coords = get_coords(trip_id, 'dumpsite')
        G = ox.graph_from_point(home_base_parking, dist=100000, network_type='drive')
        home_node = ox.distance.nearest_nodes(G, X=home_base_parking[1], Y=home_base_parking[0])
        pickup_node = ox.distance.nearest_nodes(G, X=pickup_coords[1], Y=pickup_coords[0])
        dumpsite_node = ox.distance.nearest_nodes(G, X=dumpsite_coords[1], Y=dumpsite_coords[0])
        path_home_to_pickup = nx.shortest_path(G, source=home_node, target=pickup_node,
weight='length')
        path_pickup_to_dump = nx.shortest_path(G, source=pickup_node, target=dumpsite_node,
weight='length')
        path_dump_to_home = nx.shortest_path(G, source=dumpsite_node, target=home_node,
weight='length')
        m = folium.Map(location=home_base_parking, zoom_start=12)
        folium.Marker(location=home_base_parking, popup='Home Base',
icon=folium.Icon(color='blredck', icon='home', prefix='fa')).add_to(m)
        folium.Marker(location=pickup_coords, popup=f'Pickup {trip_id}', icon=folium.Icon(color='orange',
icon='envelope', prefix='fa')).add_to(m)
        folium.Marker(location=dumpsite_coords, popup=f'Dumpsite {trip_id}',
icon=folium.Icon(color='purple', icon='trash', prefix='fa')).add_to(m)
        route_home_to_pickup = [(G.nodes[node]['y'], G.nodes[node]['x']) for node in
path_home_to_pickup]
        route_pickup_to_dump = [(G.nodes[node]['y'], G.nodes[node]['x']) for node in
path_pickup_to_dump]
        route_dump_to_home = [(G.nodes[node]['y'], G.nodes[node]['x']) for node in path_dump_to_home]
        folium.PolyLine(route_home_to_pickup, color="black", weight=2.5, opacity=1).add_to(m)
        folium.PolyLine(route_pickup_to_dump, color="green", weight=2.5, opacity=1).add_to(m)
        folium.PolyLine(route_dump_to_home, color="purple", weight=2.5, opacity=1).add_to(m)
        total_job_distance = distance_data.loc[distance_data['TRIP_ID'] == trip_id,
'total_distance(mi)'].values[0]
        display(m)
        print(f'Total Distance for the job including all 3 routes: {trip_id}: {total_job_distance:.2f} miles')
    except Exception as e:
        print(f"An error occurred: {e}")
plot_all_routes('NOV0031')
```