In [94]:
```
import pandas as pd
```

In [95]:
```
data = pd.read_csv(r"C:\Users\NITISH SINGH\Downloads\dataset-1.csv")
data.head()
```

Out[95]:

	id_1	id_2	route	moto	car	rv	bus	truck
0	829	827	1	2.05	4.14	4.14	10.1	15.2
1	829	821	4	6.63	13.26	13.26	32.4	48.5
2	829	804	7	14.41	28.92	28.92	64.7	97.0
3	829	822	6	5.90	11.81	11.81	28.8	43.2
4	829	826	9	2.87	5.81	5.81	14.2	21.2

Question 1: Car Matrix Generation

In [96]:
```
# Function to generate car matrix
def generate_car_matrix(dataframe):
    pivot_df = dataframe.pivot(index='id_1', columns='id_2', values='car').fillna(0)

    for col in pivot_df.columns:
        pivot_df.loc[pivot_df.index == col, col] = 0

    return pivot_df

# Assuming 'dataset-1.csv' contains the necessary data and 'id_1', 'id_2', 'car' are column names
# Replace 'dataset-1.csv' with the actual path to your file and read the data into a DataFrame
file_path = 'dataset-1.csv'  # Replace this with the correct file path
data = pd.read_csv(r"C:\Users\NITISH SINGH\Downloads\dataset-1.csv")

# Call the function to generate the car matrix
result_matrix = generate_car_matrix(data)
print(result_matrix)
```

```
id_2   801   802   803   804   805   806   807   808   809   821  \
id_1
801   0.00  2.80  6.00  7.70 11.70 13.40 16.90 19.60 21.00 23.52
802   2.80  0.00  3.40  5.20  9.20 10.90 14.30 17.10 18.50 20.92
803   6.00  3.40  0.00  2.00  6.00  7.70 11.10 13.90 15.30 17.72
804   7.70  5.20  2.00  0.00  4.40  6.10  9.50 12.30 13.70 16.12
805  11.70  9.20  6.00  4.40  0.00  2.00  5.40  8.20  9.60 12.02
806  13.40 10.90  7.70  6.10  2.00  0.00  3.80  6.60  8.00 10.42
807  16.90 14.30 11.10  9.50  5.40  3.80  0.00  2.90  4.30  6.82
808  19.60 17.10 13.90 12.30  8.20  6.60  2.90  0.00  1.70  4.12
809  21.00 18.50 15.30 13.70  9.60  8.00  4.30  1.70  0.00  2.92
821  23.52 20.92 17.72 16.12 12.02 10.42  6.82  4.12  2.92  0.00
822  24.67 22.07 18.87 17.27 13.17 11.57  7.97  5.27  4.07  1.80
823  26.53 23.93 20.73 19.13 15.03 13.43  9.83  7.13  5.93  3.67
824  27.92 25.32 22.12 20.52 16.42  7.80 11.22  8.52  7.32  5.06
825  29.08 26.48 23.28 21.68 17.58 15.98 12.38  9.68  8.48  6.22
826  30.87 28.27 25.07 23.47 19.37 17.77 14.17 11.47 10.27  8.01
827  32.53 29.93 26.73 25.13 21.03 19.43 15.83 13.13 11.93  9.43
829  36.32 33.72 30.52 28.92 24.82 23.22 19.62 16.92 15.72 13.26
830  38.27 35.67 32.47 30.87 26.77 25.17 21.57 18.87 17.67 15.17
831  39.24 36.64 33.44 31.84 27.74 26.14 22.54 19.84 18.64 16.15

id_2   822   823   824   825   826   827   829   830   831
id_1
801  24.67 26.53 27.92 29.08 30.87 32.53 36.32 38.27 39.24
802  22.07 23.93 25.32 26.48 28.27 29.93 33.72 35.67 36.64
803  18.87 20.73 22.12 23.28 25.07 26.73 30.52 32.47 33.44
804  17.27 19.13 20.52 21.68 23.47 25.13 28.92 30.87 31.84
805  13.17 15.03 16.42 17.58 19.37 21.03 24.82 26.77 27.74
806  11.57 13.43 14.82 15.98 17.77 19.43 23.22 25.17 26.14
807   7.97  9.83 11.22 12.38 14.17 15.83 19.62 21.57 22.54
808   5.27  7.13  8.52  9.68 11.47 13.13 16.92 18.87 19.84
809   4.07  5.93  7.32  8.48 10.27 11.93 15.72 17.67 18.64
821   1.80  3.67  5.06  6.22  8.01  9.43 13.26 15.17 16.15
822   0.00  2.21  3.60  4.76  6.55  8.00 11.81 13.74 14.68
823   2.21  0.00  1.79  2.94  4.74  6.15 10.00 11.89 12.87
824   3.60  1.79  0.00  1.71  3.50  4.92  8.77 10.66 11.64
825   4.76  2.94  1.71  0.00  2.20  3.65  7.46  9.35 10.33
826   6.55  4.74  3.50  2.20  0.00  2.05  5.81  7.71  8.69
827   8.00  6.15  4.92  3.65  2.05  0.00  4.14  6.06  7.04
829  11.81 10.00 21.40  7.46  5.81  4.14  0.00  2.38  3.36
830  13.74 11.89 10.66  0.00  7.71  6.06  2.38  0.00  1.39
831  14.68 12.87 11.64 10.33  8.69  7.04  3.36  1.39  0.00
```

Question 2: Car Type Count Calculation

In [97]:
```
def get_type_count(dataframe):
    # Adding a new categorical column 'car_type' based on values of the column 'car'
    dataframe['car_type'] = pd.cut(dataframe['car'],
                                   bins=[float('-inf'), 15, 25, float('inf')],
                                   labels=['low', 'medium', 'high'],
                                   right=False)

    # Calculating the count of occurrences for each 'car_type' category
    type_counts = dataframe['car_type'].value_counts().to_dict()

    # Sort the dictionary alphabetically based on keys
    sorted_type_counts = dict(sorted(type_counts.items()))

    return sorted_type_counts

# Assuming 'dataset-1.csv' contains the necessary data and 'car' is a column name
# Replace 'dataset-1.csv' with the actual path to your file and read the data into a DataFrame
file_path = 'dataset-1.csv'  # Replace this with the correct file path
data = pd.read_csv(r"C:\Users\NITISH SINGH\Downloads\dataset-1.csv")

# Call the function to get the count of occurrences for each car_type category
result_dict = get_type_count(data)
print(result_dict)
```

```
{'high': 56, 'low': 196, 'medium': 89}
```

Question 3: Bus Count Index Retrieval

In [98]:
```
def get_bus_indexes(dataframe):
    # Calculate the mean value of the 'bus' column
    mean_bus_value = dataframe['bus'].mean()

    # Identify indices where the 'bus' values are greater than twice the mean value
    bus_indexes = dataframe[dataframe['bus'] > 2 * mean_bus_value].index.tolist()

    # Sort the indices in ascending order
    bus_indexes.sort()

    return bus_indexes

# Assuming 'dataset-1.csv' contains the necessary data and 'bus' is a column name
# Replace 'dataset-1.csv' with the actual path to your file and read the data into a DataFrame
file_path = 'dataset-1.csv'  # Replace this with the correct file path
data = pd.read_csv(r"C:\Users\NITISH SINGH\Downloads\dataset-1.csv")

# Call the function to get the indices where 'bus' values are greater than twice the mean
result_indices = get_bus_indexes(data)
print(result_indices)
```

```
[2, 7, 12, 17, 25, 30, 54, 64, 70, 97, 144, 145, 149, 154, 160, 201, 206, 210, 215, 234, 235, 245, 250, 309, 314, 319, 322, 323, 334, 340]
```

Question 4: Route Filtering

In [99]:
```
data = pd.read_csv(r"C:\Users\NITISH SINGH\Downloads\dataset-1.csv")
data.head()
def filter_routes(dataframe):
    # Grouping by 'route' column and calculating the average of 'truck' column for each route
    route_avg_truck = dataframe.groupby('route')['truck'].mean()

    # Filtering routes where the average of 'truck' column values is greater than 7
    filtered_routes = route_avg_truck[route_avg_truck > 7].index.tolist()

    # Sorting the routes in ascending order
    filtered_routes.sort()

    return filtered_routes
# Call the function to filter routes where the average of 'truck' column values is greater than 7
result_routes = filter_routes(data)
print(result_routes)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Question 5: Matrix Value Modification

In [100]:
```
import pandas as pd
```

```
# Assuming you have a CSV file named 'dataset-1.csv' with appropriate data
file_path = r"C:\Users\NITISH SINGH\Downloads\dataset-1.csv"

def modify_dataframe(dataframe):
    # Apply the specified logic to modify values in the DataFrame
    modified_dataframe = dataframe.applymap(
        lambda x: x * 0.75 if x > 20 else x * 1.25
    )

    # Round the values to 1 decimal place
    modified_dataframe = modified_dataframe.round(1)

    return modified_dataframe

# Read the CSV file into a DataFrame
modified_dataframe = pd.read_csv(r"C:\Users\NITISH SINGH\Downloads\dataset-1.csv")

# Call the function with the DataFrame read from the CSV file
result = modify_dataframe(modified_dataframe)
print(result)
```

```
     id_1   id_2  route  moto   car    rv   bus  truck
0    621.8  620.2    1.2   2.6   5.2   5.2  12.6   19.0
1    621.8  615.8    5.0   8.3  16.6  16.6  24.3   36.4
2    621.8  603.0    8.8  18.0  21.7  21.7  48.5   72.8
3    621.8  616.5    7.5   7.4  14.8  14.8  21.6   32.4
4    621.8  619.5   11.2   3.6   7.3   7.3  17.8   15.9
..     ...    ...    ...   ...   ...   ...   ...    ...
336  602.2  601.5    3.8   2.1   4.2   4.2   8.6   12.9
337  602.2  603.8    5.0   3.8   7.5   7.5  15.0   22.4
338  602.2  618.8    3.8  14.5  17.5  17.5  37.6   56.4
339  602.2  604.5   11.2   4.8   9.6   9.6  19.1   17.2
340  602.2  622.5    1.2  20.2  24.4  24.4  54.4   81.6

[341 rows x 8 columns]
```

Question 6: Time Check

In [101]:
```
data = pd.read_csv(r"C:\Users\NITISH SINGH\Downloads\dataset-2.csv")
data.head
```

Out[101]:
```
<bound method NDFrame.head of            id        name     id_2  startDay startTime     endDay   endTime  \
0     1040000  Montgomery       -1    Monday  05:00:00  Wednesday  10:00:00
1     1040010       Black       -1    Monday  10:00:00     Friday  15:00:00
2     1040020     Emerald       -1  Thursday  15:00:00     Friday  19:00:00
3     1040030       Foley       -1    Monday  19:00:00     Friday  23:59:59
4     1050000    Whittier  1050001  Saturday  00:00:00     Sunday  23:59:59
...       ...         ...      ...       ...       ...        ...       ...
39509 1031012     Baldwin  1031030    Monday  19:00:00     Friday  23:59:59
39510 1031012     Baldwin  1031032  Saturday  00:00:00     Sunday  23:59:59
39511 1031014    Thickson  1031016  Saturday  00:00:00     Sunday  23:59:59
39512 1031014    Thickson  1031018    Monday  05:00:00  Wednesday  10:00:00
39513 1031014    Thickson  1031020    Monday  10:00:00     Friday  15:00:00

       able2Hov2  able2Hov3  able3Hov2  able3Hov3  able5Hov2  able5Hov3  \
0            3.0        3.0       -1.0         -1          3          3
1            6.0        6.0       -1.0         -1          6          6
2            3.0        3.0       -1.0         -1          3          3
3            6.0        6.0       -1.0         -1          6          6
4            6.0        6.0        NaN         -1          6          6
...          ...        ...        ...        ...        ...        ...
39509       11.0       11.0        4.0          4         11         11
39510       11.0       11.0        4.0          4         11         11
39511       11.0       11.0        4.0          4         11         11
39512        8.0        8.0        4.0          4          8          8
39513       11.0       11.0        4.0          4         11         11

       able4Hov2  able4Hov3
0              3          3
1              6          6
2              3          3
3              6          6
4              6          6
...          ...        ...
39509         11         11
39510         11         11
39511         11         11
39512          8          8
39513         11         11

[39514 rows x 15 columns]>
```

In [102]: data.isnull().sum()

Out[102]:
```
id            0
name          0
id_2          0
startDay      0
startTime     0
endDay        0
endTime       0
able2Hov2    1806
able2Hov3    1805
able3Hov2    1805
able3Hov3       0
able5Hov2       0
able5Hov3       0
able4Hov2       0
able4Hov3       0
dtype: int64
```

In [103]:
```
data = data.dropna()
```

In [104]:
```
data.dropna()
```

Out[104]:

	id	name	id_2	startDay	startTime	endDay	endTime	able2Hov2	able2Hov3	able3Hov2	able3Hov3	able5Hov2	able5Ho
0	1040000	Montgomery	-1	Monday	05:00:00	Wednesday	10:00:00	3.0	3.0	-1.0	-1	3	
1	1040010	Black	-1	Monday	10:00:00	Friday	15:00:00	6.0	6.0	-1.0	-1	6	
2	1040020	Emerald	-1	Thursday	15:00:00	Friday	19:00:00	3.0	3.0	-1.0	-1	3	
3	1040030	Foley	-1	Monday	19:00:00	Friday	23:59:59	6.0	6.0	-1.0	-1	6	
7	1200000	SR	1200004	Monday	10:00:00	Friday	15:00:00	6.0	6.0	-1.0	-1	6	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
39509	1031012	Baldwin	1031030	Monday	19:00:00	Friday	23:59:59	11.0	11.0	4.0	4	11	
39510	1031012	Baldwin	1031032	Saturday	00:00:00	Sunday	23:59:59	11.0	11.0	4.0	4	11	
39511	1031014	Thickson	1031016	Saturday	00:00:00	Sunday	23:59:59	11.0	11.0	4.0	4	11	
39512	1031014	Thickson	1031018	Monday	05:00:00	Wednesday	10:00:00	8.0	8.0	4.0	4	8	
39513	1031014	Thickson	1031020	Monday	10:00:00	Friday	15:00:00	11.0	11.0	4.0	4	11	

34098 rows × 15 columns

In [105]:
```
data.isna().sum()
```

Out[105]:
```
id            0
name          0
id_2          0
startDay      0
startTime     0
endDay        0
endTime       0
able2Hov2     0
able2Hov3     0
able3Hov2     0
able3Hov3     0
able5Hov2     0
able5Hov3     0
able4Hov2     0
able4Hov3     0
dtype: int64
```

In [106]:
```
import pandas as pd

def check_timestamps(dataframe):
    # Combine date and time columns into single datetime columns with explicit format
    dataframe['start_datetime'] = pd.to_datetime(dataframe['startDay'] + ' ' + dataframe['startTime'], format='%Y-%m-%d %H:%M:%S', errors='coerce')
    dataframe['end_datetime'] = pd.to_datetime(dataframe['endDay'] + ' ' + dataframe['endTime'], format='%Y-%m-%d %H:%M:%S', errors='coerce')

    # Remove rows with NaT (Not a Timestamp) values resulting from conversion errors
    dataframe.dropna(subset=['start_datetime', 'end_datetime'], inplace=True)

    # Group by (id, id_2) pairs and get the minimum start time and maximum end time
    grouped = dataframe.groupby(['id', 'id_2']).agg(
        start_time=('start_datetime', 'min'),
        end_time=('end_datetime', 'max')
    )

    # Calculate time duration for each (id, id_2) pair
    grouped['duration'] = grouped['end_time'] - grouped['start_time']

    # Define the criteria for incorrect timestamps
    incorrect_timestamps = (
```

```
    (grouped['duration'] < pd.Timedelta(days=7))  # Span less than 7 days
    | (grouped['start_time'].dt.weekday.min() != 0)  # Does not start on Monday
    | (grouped['end_time'].dt.weekday.max() != 6)  # Does not end on Sunday
    | (grouped['start_time'].dt.hour.min() != 0)  # Does not start at 12:00:00 AM
    | (grouped['end_time'].dt.hour.max() != 23)  # Does not end at 11:59:59 PM
  )

  return incorrect_timestamps

# Load dataset-2.csv into a DataFrame
file_path = r"C:\Users\NITISH SINGH\Downloads\dataset-2.csv"
dataset = pd.read_csv(file_path)

# Call the function with the dataset to get the boolean series indicating incorrect timestamps for each (id, id_2) pair
incorrect_timestamps_series = check_timestamps(dataset)
print(incorrect_timestamps_series)
```

```
Series([], Name: duration, dtype: bool)
```

TASK-2

Question 1: Distance Matrix Calculation

In [107]:
```
data = pd.read_csv(r"C:\Users\NITISH SINGH\Downloads\dataset-3.csv")
data.head
```

Out[107]:
```
<bound method NDFrame.head of     id_start   id_end  distance
0    1001400  1001402       9.7
1    1001402  1001404      20.2
2    1001404  1001406      16.0
3    1001406  1001408      21.7
4    1001408  1001410      11.1
5    1001410  1001412      15.6
6    1001412  1001414      18.2
7    1001414  1001416      13.2
8    1001416  1001418      13.6
9    1001418  1001420      12.9
10   1001420  1001422       9.6
11   1001422  1001424      11.4
12   1001424  1001426      18.6
13   1001426  1001428      15.8
14   1001428  1001430       8.6
15   1001430  1001432       9.0
16   1001432  1001434       7.9
17   1001434  1001436       4.0
18   1001436  1001438       9.0
19   1001436  1001437       5.0
20   1001438  1001437       4.0
21   1001438  1001440      10.0
22   1001440  1001442       3.9
23   1001442  1001488       4.5
24   1001488  1004356       4.0
25   1004356  1004354       2.0
26   1004354  1004355       2.0
27   1004355  1001444       0.7
28   1001444  1001446       6.6
29   1001446  1001448       9.6
30   1001448  1001450      15.7
31   1001450  1001452       9.9
32   1001452  1001454      11.3
33   1001454  1001456      13.6
34   1001456  1001458       8.9
35   1001458  1001460       5.1
36   1001460  1001461      12.8
37   1001460  1001462      17.9
38   1001461  1001462       5.1
39   1001462  1001464      26.7
40   1001464  1001466       8.5
41   1001466  1001468      10.7
42   1001468  1001470      10.6
43   1001470  1001472      16.0>
```

In [108]:
```
def calculate_distance_matrix(file_path):
  # Load the dataset
  dataset = pd.read_csv( r"C:\Users\NITISH SINGH\Downloads\dataset-3.csv")

  # Extract unique IDs (toll locations) from the dataset
  unique_ids = sorted(set(dataset['id_start']) | set(dataset['id_end']))

  # Initialize a dictionary to store distances
```

```
    distances = {(start, end): 0 for start in unique_ids for end in unique_ids}

    # Iterate through the dataset to compute distances and fill the dictionary
    for _, row in dataset.iterrows():
        start = row['id_start']
        end = row['id_end']
        distance = row['distance']

        # Accumulate distances for each pair of toll locations
        distances[(start, end)] += distance
        distances[(end, start)] += distance  # Accounting for bidirectional distances

    # Create an empty DataFrame for the distance matrix
    distance_matrix = pd.DataFrame(index=unique_ids, columns=unique_ids)

    # Fill the DataFrame with cumulative distances
    for start in unique_ids:
        for end in unique_ids:
            distance_matrix.loc[start, end] = distances[(start, end)]

    # Set diagonal values to 0
    distance_matrix.values[[range(len(unique_ids))]*2] = 0

    return distance_matrix.astype(float)  # Convert distances to float

# Provide the file path to dataset-3.csv and call the function
file_path = 'path/to/dataset-3.csv'  # Replace with the actual file path
resulting_distance_matrix = calculate_distance_matrix(file_path)

# Display the resulting distance matrix
print(resulting_distance_matrix)
```

```
         1001400  1001402  1001404  1001406  1001408  1001410  1001412  \
1001400      0.0      9.7      0.0      0.0      0.0      0.0      0.0
1001402      9.7      0.0     20.2      0.0      0.0      0.0      0.0
1001404      0.0     20.2      0.0     16.0      0.0      0.0      0.0
1001406      0.0      0.0     16.0      0.0     21.7      0.0      0.0
1001408      0.0      0.0      0.0     21.7      0.0     11.1      0.0
1001410      0.0      0.0      0.0      0.0     11.1      0.0     15.6
1001412      0.0      0.0      0.0      0.0      0.0     15.6      0.0
1001414      0.0      0.0      0.0      0.0      0.0      0.0     18.2
1001416      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001418      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001420      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001422      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001424      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001426      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001428      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001430      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001432      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001434      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001436      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001437      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001438      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001440      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001442      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001444      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001446      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001448      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001450      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001452      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001454      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001456      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001458      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001460      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001461      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001462      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001464      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001466      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001468      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001470      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001472      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1001488      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1004354      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1004355      0.0      0.0      0.0      0.0      0.0      0.0      0.0
1004356      0.0      0.0      0.0      0.0      0.0      0.0      0.0

         1001414  1001416  1001418  ...  1001462  1001464  1001466  1001468  \
1001400      0.0      0.0      0.0  ...      0.0      0.0      0.0      0.0
1001402      0.0      0.0      0.0  ...      0.0      0.0      0.0      0.0
1001404      0.0      0.0      0.0  ...      0.0      0.0      0.0      0.0
1001406      0.0      0.0      0.0  ...      0.0      0.0      0.0      0.0
1001408      0.0      0.0      0.0  ...      0.0      0.0      0.0      0.0
```

1001410	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001412	18.2	0.0	0.0	...	0.0	0.0	0.0	0.0
1001414	0.0	13.2	0.0	...	0.0	0.0	0.0	0.0
1001416	13.2	0.0	13.6	...	0.0	0.0	0.0	0.0
1001418	0.0	13.6	0.0	...	0.0	0.0	0.0	0.0
1001420	0.0	0.0	12.9	...	0.0	0.0	0.0	0.0
1001422	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001424	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001426	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001428	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001430	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001432	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001434	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001436	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001437	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001438	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001440	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001442	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001444	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001446	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001448	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001450	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001452	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001454	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001456	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001458	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001460	0.0	0.0	0.0	...	17.9	0.0	0.0	0.0
1001461	0.0	0.0	0.0	...	5.1	0.0	0.0	0.0
1001462	0.0	0.0	0.0	...	0.0	26.7	0.0	0.0
1001464	0.0	0.0	0.0	...	26.7	0.0	8.5	0.0
1001466	0.0	0.0	0.0	...	0.0	8.5	0.0	10.7
1001468	0.0	0.0	0.0	...	0.0	0.0	10.7	0.0
1001470	0.0	0.0	0.0	...	0.0	0.0	0.0	10.6
1001472	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001488	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1004354	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1004355	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1004356	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

	1001470	1001472	1001488	1004354	1004355	1004356
1001400	0.0	0.0	0.0	0.0	0.0	0.0
1001402	0.0	0.0	0.0	0.0	0.0	0.0
1001404	0.0	0.0	0.0	0.0	0.0	0.0
1001406	0.0	0.0	0.0	0.0	0.0	0.0
1001408	0.0	0.0	0.0	0.0	0.0	0.0
1001410	0.0	0.0	0.0	0.0	0.0	0.0
1001412	0.0	0.0	0.0	0.0	0.0	0.0
1001414	0.0	0.0	0.0	0.0	0.0	0.0
1001416	0.0	0.0	0.0	0.0	0.0	0.0
1001418	0.0	0.0	0.0	0.0	0.0	0.0
1001420	0.0	0.0	0.0	0.0	0.0	0.0
1001422	0.0	0.0	0.0	0.0	0.0	0.0
1001424	0.0	0.0	0.0	0.0	0.0	0.0
1001426	0.0	0.0	0.0	0.0	0.0	0.0
1001428	0.0	0.0	0.0	0.0	0.0	0.0
1001430	0.0	0.0	0.0	0.0	0.0	0.0
1001432	0.0	0.0	0.0	0.0	0.0	0.0
1001434	0.0	0.0	0.0	0.0	0.0	0.0
1001436	0.0	0.0	0.0	0.0	0.0	0.0
1001437	0.0	0.0	0.0	0.0	0.0	0.0
1001438	0.0	0.0	0.0	0.0	0.0	0.0
1001440	0.0	0.0	0.0	0.0	0.0	0.0
1001442	0.0	0.0	4.5	0.0	0.0	0.0
1001444	0.0	0.0	0.0	0.0	0.7	0.0
1001446	0.0	0.0	0.0	0.0	0.0	0.0
1001448	0.0	0.0	0.0	0.0	0.0	0.0
1001450	0.0	0.0	0.0	0.0	0.0	0.0
1001452	0.0	0.0	0.0	0.0	0.0	0.0
1001454	0.0	0.0	0.0	0.0	0.0	0.0
1001456	0.0	0.0	0.0	0.0	0.0	0.0
1001458	0.0	0.0	0.0	0.0	0.0	0.0
1001460	0.0	0.0	0.0	0.0	0.0	0.0
1001461	0.0	0.0	0.0	0.0	0.0	0.0
1001462	0.0	0.0	0.0	0.0	0.0	0.0
1001464	0.0	0.0	0.0	0.0	0.0	0.0
1001466	0.0	0.0	0.0	0.0	0.0	0.0
1001468	10.6	0.0	0.0	0.0	0.0	0.0
1001470	0.0	16.0	0.0	0.0	0.0	0.0
1001472	16.0	0.0	0.0	0.0	0.0	0.0
1001488	0.0	0.0	0.0	0.0	0.0	4.0
1004354	0.0	0.0	0.0	0.0	2.0	2.0
1004355	0.0	0.0	0.0	2.0	0.0	0.0
1004356	0.0	0.0	4.0	2.0	0.0	0.0

1004456 0.0 0.0 4.0 2.0 0.0 0.0

[43 rows x 43 columns]
C:\Users\NITISH SINGH\AppData\Local\Temp\ipykernel_2976\3383168958.py:30: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in a n error or a different result.
distance_matrix.values[[range(len(unique_ids))]*2] = 0

Question 2: Unroll Distance Matrix

In [112]:
```
import pandas as pd

def unroll_distance_matrix(calculate_distance_matrix):
    # Extract the IDs from the index of the distance matrix
    ids = calculate_distance_matrix.index.tolist()

    # Initialize lists to store id_start, id_end, and distance values
    id_start = []
    id_end = []
    distance = []

    # Iterate through the distance matrix to generate combinations and distances
    for i in range(len(ids)):
        for j in range(len(ids)):
            if i != j:
                id_start.append(ids[i])
                id_end.append(ids[j])
                distance.append(calculate_distance_matrix.iloc[i, j])

    # Create a DataFrame with id_start, id_end, and distance columns
    unrolled_distance_df = pd.DataFrame({
        'id_start': id_start,
        'id_end': id_end,
        'distance': distance
    })

    return unrolled_distance_df

# Replace distance_matrix with the actual DataFrame from Question 1
# For example:
# distance_matrix = calculate_distance_matrix(dataset_3)
# where dataset_3 is the DataFrame generated from the previous function
resulting_unrolled_distance_df = calculate_distance_matrix(calculate_distance_matrix)

# Display the resulting unrolled distance DataFrame
print(resulting_unrolled_distance_df)
```

	1001400	1001402	1001404	1001406	1001408	1001410	1001412
1001400	0.0	9.7	0.0	0.0	0.0	0.0	0.0
1001402	9.7	0.0	20.2	0.0	0.0	0.0	0.0
1001404	0.0	20.2	0.0	16.0	0.0	0.0	0.0
1001406	0.0	0.0	16.0	0.0	21.7	0.0	0.0
1001408	0.0	0.0	0.0	21.7	0.0	11.1	0.0
1001410	0.0	0.0	0.0	0.0	11.1	0.0	15.6
1001412	0.0	0.0	0.0	0.0	0.0	15.6	0.0
1001414	0.0	0.0	0.0	0.0	0.0	0.0	18.2
1001416	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001418	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001420	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001422	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001424	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001426	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001428	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001430	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001432	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001434	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001436	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001437	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001438	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001440	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001442	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001444	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001446	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001448	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001450	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001452	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001454	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001456	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001458	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001460	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001461	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001462	0.0	0.0	0.0	0.0	0.0	0.0	0.0

1001464	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001466	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001468	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001470	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001472	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1001488	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1004354	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1004355	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1004356	0.0	0.0	0.0	0.0	0.0	0.0	0.0

	1001414	1001416	1001418	...	1001462	1001464	1001466	1001468 \
1001400	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001402	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001404	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001406	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001408	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001410	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001412	18.2	0.0	0.0	...	0.0	0.0	0.0	0.0
1001414	0.0	13.2	0.0	...	0.0	0.0	0.0	0.0
1001416	13.2	0.0	13.6	...	0.0	0.0	0.0	0.0
1001418	0.0	13.6	0.0	...	0.0	0.0	0.0	0.0
1001420	0.0	0.0	12.9	...	0.0	0.0	0.0	0.0
1001422	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001424	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001426	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001428	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001430	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001432	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001434	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001436	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001437	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001438	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001440	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001442	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001444	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001446	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001448	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001450	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001452	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001454	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001456	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001458	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001460	0.0	0.0	0.0	...	17.9	0.0	0.0	0.0
1001461	0.0	0.0	0.0	...	5.1	0.0	0.0	0.0
1001462	0.0	0.0	0.0	...	0.0	26.7	0.0	0.0
1001464	0.0	0.0	0.0	...	26.7	0.0	8.5	0.0
1001466	0.0	0.0	0.0	...	0.0	8.5	0.0	10.7
1001468	0.0	0.0	0.0	...	0.0	0.0	10.7	0.0
1001470	0.0	0.0	0.0	...	0.0	0.0	0.0	10.6
1001472	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1001488	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1004354	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1004355	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0
1004356	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0

	1001470	1001472	1001488	1004354	1004355	1004356
1001400	0.0	0.0	0.0	0.0	0.0	0.0
1001402	0.0	0.0	0.0	0.0	0.0	0.0
1001404	0.0	0.0	0.0	0.0	0.0	0.0
1001406	0.0	0.0	0.0	0.0	0.0	0.0
1001408	0.0	0.0	0.0	0.0	0.0	0.0
1001410	0.0	0.0	0.0	0.0	0.0	0.0
1001412	0.0	0.0	0.0	0.0	0.0	0.0
1001414	0.0	0.0	0.0	0.0	0.0	0.0
1001416	0.0	0.0	0.0	0.0	0.0	0.0
1001418	0.0	0.0	0.0	0.0	0.0	0.0
1001420	0.0	0.0	0.0	0.0	0.0	0.0
1001422	0.0	0.0	0.0	0.0	0.0	0.0
1001424	0.0	0.0	0.0	0.0	0.0	0.0
1001426	0.0	0.0	0.0	0.0	0.0	0.0
1001428	0.0	0.0	0.0	0.0	0.0	0.0
1001430	0.0	0.0	0.0	0.0	0.0	0.0
1001432	0.0	0.0	0.0	0.0	0.0	0.0
1001434	0.0	0.0	0.0	0.0	0.0	0.0
1001436	0.0	0.0	0.0	0.0	0.0	0.0
1001437	0.0	0.0	0.0	0.0	0.0	0.0
1001438	0.0	0.0	0.0	0.0	0.0	0.0
1001440	0.0	0.0	0.0	0.0	0.0	0.0
1001442	0.0	0.0	4.5	0.0	0.0	0.0
1001444	0.0	0.0	0.0	0.0	0.7	0.0
1001446	0.0	0.0	0.0	0.0	0.0	0.0
1001448	0.0	0.0	0.0	0.0	0.0	0.0
1001450	0.0	0.0	0.0	0.0	0.0	0.0

1001452	0.0	0.0	0.0	0.0	0.0	0.0
1001454	0.0	0.0	0.0	0.0	0.0	0.0
1001456	0.0	0.0	0.0	0.0	0.0	0.0
1001458	0.0	0.0	0.0	0.0	0.0	0.0
1001460	0.0	0.0	0.0	0.0	0.0	0.0
1001461	0.0	0.0	0.0	0.0	0.0	0.0
1001462	0.0	0.0	0.0	0.0	0.0	0.0
1001464	0.0	0.0	0.0	0.0	0.0	0.0
1001466	0.0	0.0	0.0	0.0	0.0	0.0
1001468	10.6	0.0	0.0	0.0	0.0	0.0
1001470	0.0	16.0	0.0	0.0	0.0	0.0
1001472	16.0	0.0	0.0	0.0	0.0	0.0
1001488	0.0	0.0	0.0	0.0	0.0	4.0
1004354	0.0	0.0	0.0	0.0	2.0	2.0
1004355	0.0	0.0	0.0	2.0	0.0	0.0
1004356	0.0	0.0	4.0	2.0	0.0	0.0

[43 rows x 43 columns]

```
C:\Users\NITISH SINGH\AppData\Local\Temp\ipykernel_2976\3383168958.py:30: FutureWarning: Using a non-tuple sequence for multidimensional indexing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the future this will be interpreted as an array index, `arr[np.array(seq)]`, which will result either in a n error or a different result.
  distance_matrix.values[[range(len(unique_ids))]*2] = 0
```

Question 3: Finding IDs within Percentage Threshold

In [122]:
```
import pandas as pd

def find_ids_within_ten_percentage_threshold(distance_df, reference_value):
    # Filter the DataFrame for the reference value and calculate its average distance
    avg_distance = distance_df[distance_df['id_start'] == reference_value]['distance'].mean()

    # Calculate the 10% threshold
    threshold = 0.1 * avg_distance

    # Calculate the lower and upper bounds
    lower_bound = avg_distance - threshold
    upper_bound = avg_distance + threshold

    # Filter the DataFrame for values within the threshold range
    within_threshold = distance_df[(distance_df['id_start'] <= reference_value) &
                                   (distance_df['distance'] >= lower_bound) &
                                   (distance_df['distance'] <= upper_bound)]

    # Get unique values of id_start that satisfy the threshold condition
    values_within_threshold = sorted(within_threshold['id_start'].unique())

    return values_within_threshold
# Assuming 'distance_df' is the DataFrame generated from Question 2
# Replace 'distance_df' with your actual DataFrame

# Choose a reference value
reference_value = 123  # Replace with the desired reference value


# Display the sorted list of values within the 10% threshold of the reference value's average distance
print(result)
# Assuming 'resulting_unrolled_distance_df' is the DataFrame generated from the unroll_distance_matrix function
# Replace 'resulting_unrolled_distance_df' with your actual DataFrame

# Choose a reference value
reference_value = 123  # Replace with the desired reference value
# Display the sorted list of values within the 10% threshold of the reference value's average distance
print(result)
```

```
     id_1   id_2  route  moto   car    rv   bus  truck
0   621.8  620.2    1.2   2.6   5.2   5.2  12.6   19.0
1   621.8  615.8    5.0   8.3  16.6  16.6  24.3   36.4
2   621.8  603.0    8.8  18.0  21.7  21.7  48.5   72.8
3   621.8  616.5    7.5   7.4  14.8  14.8  21.6   32.4
4   621.8  619.5   11.2   3.6   7.3   7.3  17.8   15.9
..    ...    ...    ...   ...   ...   ...   ...    ...
336  602.2  601.5    3.8   2.1   4.2   4.2   8.6   12.9
337  602.2  603.8    5.0   3.8   7.5   7.5  15.0   22.4
338  602.2  618.8    3.8  14.5  17.5  17.5  37.6   56.4
339  602.2  604.5   11.2   4.8   9.6   9.6  19.1   17.2
340  602.2  622.5    1.2  20.2  24.4  24.4  54.4   81.6

[341 rows x 8 columns]
```


```
     id_1   id_2  route  moto   car    rv   bus  truck
0   621.8  620.2    1.2   2.6   5.2   5.2  12.6   19.0
1   621.8  615.8    5.0   8.3  16.6  16.6  24.3   36.4
2   621.8  603.0    8.8  18.0  21.7  21.7  48.5   72.8
3   621.8  616.5    7.5   7.4  14.8  14.8  21.6   32.4
4   621.8  619.5   11.2   3.6   7.3   7.3  17.8   15.9
..    ...    ...    ...   ...   ...   ...   ...    ...
336  602.2  601.5    3.8   2.1   4.2   4.2   8.6   12.9
337  602.2  603.8    5.0   3.8   7.5   7.5  15.0   22.4
338  602.2  618.8    3.8  14.5  17.5  17.5  37.6   56.4
339  602.2  604.5   11.2   4.8   9.6   9.6  19.1   17.2
340  602.2  622.5    1.2  20.2  24.4  24.4  54.4   81.6

[341 rows x 8 columns]
```


Question 4: Calculate Toll Rate

In [ ]: