In [1]:
```
import pandas as pd
import numpy as np

def calculate_distance_matrix(df: pd.DataFrame) -> pd.DataFrame:

    unique_ids = pd.unique(df[['id_start', 'id_end']].values.ravel('K'))

    distance_matrix = pd.DataFrame(np.nan, index=unique_ids, columns=unique_i

    for _, row in df.iterrows():
        distance_matrix.at[row['id_start'], row['id_end']] = row['distance']
        distance_matrix.at[row['id_end'], row['id_start']] = row['distance']

    np.fill_diagonal(distance_matrix.values, 0)

    for k in unique_ids:
        for i in unique_ids:
            for j in unique_ids:
                if pd.notna(distance_matrix.at[i, k]) and pd.notna(distance_m
                    new_distance = distance_matrix.at[i, k] + distance_matrix
                    if pd.isna(distance_matrix.at[i, j]) or new_distance < di
                        distance_matrix.at[i, j] = new_distance

    return distance_matrix

file_path = r'C:\Users\Reddy\Downloads\dataset-2.csv'
df = pd.read_csv(file_path)

distance_matrix = calculate_distance_matrix(df)

print(distance_matrix)
```

```
         1001400  1001402  1001404  1001406  1001408  1001410  1001412  \
1001400      0.0      9.7     29.9     45.9     67.6     78.7     94.3
1001402      9.7      0.0     20.2     36.2     57.9     69.0     84.6
1001404     29.9     20.2      0.0     16.0     37.7     48.8     64.4
1001406     45.9     36.2     16.0      0.0     21.7     32.8     48.4
1001408     67.6     57.9     37.7     21.7      0.0     11.1     26.7
1001410     78.7     69.0     48.8     32.8     11.1      0.0     15.6
1001412     94.3     84.6     64.4     48.4     26.7     15.6      0.0
1001414    112.5    102.8     82.6     66.6     44.9     33.8     18.2
1001416    125.7    116.0     95.8     79.8     58.1     47.0     31.4
1001418    139.3    129.6    109.4     93.4     71.7     60.6     45.0
1001420    152.2    142.5    122.3    106.3     84.6     73.5     57.9
1001422    161.8    152.1    131.9    115.9     94.2     83.1     67.5
1001424    173.2    163.5    143.3    127.3    105.6     94.5     78.9
1001426    191.8    182.1    161.9    145.9    124.2    113.1     97.5
1001428    207.6    197.9    177.7    161.7    140.0    128.9    113.3
1001430    216.2    206.5    186.3    170.3    148.6    137.5    121.9
1001432    225.2    215.5    195.3    179.3    157.6    146.5    130.9
1001434    233.1    223.4    203.2    187.2    165.5    154.4    138.8
```

In [2]:
```
import pandas as pd

def unroll_distance_matrix(df: pd.DataFrame) -> pd.DataFrame:

    unrolled_data = []

    for id_start in df.index:
        for id_end in df.columns:
            distance = df.at[id_start, id_end]
            if id_start != id_end:
                unrolled_data.append({'id_start': id_start, 'id_end': id_end,

    unrolled_df = pd.DataFrame(unrolled_data)

    return unrolled_df

unrolled_df = unroll_distance_matrix(distance_matrix)
print(unrolled_df)
```

```
      id_start   id_end  distance
0      1001400  1001402       9.7
1      1001400  1001404      29.9
2      1001400  1001406      45.9
3      1001400  1001408      67.6
4      1001400  1001410      78.7
...        ...      ...       ...
1801   1001472  1001464      45.8
1802   1001472  1001466      37.3
1803   1001472  1001468      26.6
1804   1001472  1001470      16.0
1805   1001472  1001437     202.2

[1806 rows x 3 columns]
```

In [3]:
```
import pandas as pd

def find_ids_within_ten_percentage_threshold(df: pd.DataFrame, reference_id:

    avg_distance_ref = df[df['id_start'] == reference_id]['distance'].mean()

    if pd.isna(avg_distance_ref):
        return pd.DataFrame(columns=['id_start', 'average_distance'])

    lower_bound = avg_distance_ref * 0.90
    upper_bound = avg_distance_ref * 1.10

    avg_distances = df.groupby('id_start')['distance'].mean().reset_index()

    filtered_ids = avg_distances[(avg_distances['distance'] >= lower_bound) &
                                 (avg_distances['distance'] <= upper_bound)]

    filtered_ids = filtered_ids.sort_values(by='distance', ascending=True)

    return filtered_ids

reference_id = 1001400  # Example reference ID
result_df = find_ids_within_ten_percentage_threshold(unrolled_df, reference_i
print(result_df)
```

```
   id_start    distance
1   1001402  234.526190
0   1001400  243.995238
```

In [4]:
```
import pandas as pd

def calculate_toll_rate(df: pd.DataFrame) -> pd.DataFrame:

    rate_coefficients = {
        'moto': 0.8,
        'car': 1.2,
        'rv': 1.5,
        'bus': 2.2,
        'truck': 3.6
    }

    for vehicle_type, coefficient in rate_coefficients.items():
        df[vehicle_type] = df['distance'] * coefficient

    return df

toll_rates_df = calculate_toll_rate(unrolled_df)
print(toll_rates_df)
```

```
      id_start   id_end  distance    moto     car      rv     bus   truck
0      1001400  1001402       9.7    7.76   11.64   14.55   21.34   34.92
1      1001400  1001404      29.9   23.92   35.88   44.85   65.78  107.64
2      1001400  1001406      45.9   36.72   55.08   68.85  100.98  165.24
3      1001400  1001408      67.6   54.08   81.12  101.40  148.72  243.36
4      1001400  1001410      78.7   62.96   94.44  118.05  173.14  283.32
...        ...      ...       ...     ...     ...     ...     ...     ...
1801   1001472  1001464      45.8   36.64   54.96   68.70  100.76  164.88
1802   1001472  1001466      37.3   29.84   44.76   55.95   82.06  134.28
1803   1001472  1001468      26.6   21.28   31.92   39.90   58.52   95.76
1804   1001472  1001470      16.0   12.80   19.20   24.00   35.20   57.60
1805   1001472  1001437     202.2  161.76  242.64  303.30  444.84  727.92

[1806 rows x 8 columns]
```

In [11]:
```
import pandas as pd
from datetime import time

def calculate_time_based_toll_rates(df: pd.DataFrame) -> pd.DataFrame:
    """
    Calculate time-based toll rates for different time intervals within a day

    Args:
        df (pandas.DataFrame): Input DataFrame containing toll rates.

    Returns:
        pandas.DataFrame: DataFrame with time-based toll rates.
    """
    # Define discount factors
    weekday_discount_factors = {
        'morning': 0.8,   # 00:00 to 10:00
        'afternoon': 1.2, # 10:00 to 18:00
        'evening': 0.8    # 18:00 to 23:59
    }
    weekend_discount_factor = 0.7

    # Days of the week
    days_of_week = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', '

    # Prepare a list to collect new rows
    new_rows = []

    # Unique (id_start, id_end) pairs
    unique_pairs = df[['id_start', 'id_end', 'distance']].drop_duplicates()

    # Generate time-based toll rates for weekdays and weekends
    for index, row in unique_pairs.iterrows():
        id_start = row['id_start']
        id_end = row['id_end']
        distance = row['distance']

        # Weekdays (Monday to Friday)
        for day in days_of_week[:5]:  # Monday to Friday
            # Morning
            new_rows.append({
                'id_start': id_start,
                'id_end': id_end,
                'distance': distance,
                'start_day': day,
                'start_time': time(0, 0),   # 12:00 AM
                'end_day': day,
                'end_time': time(10, 0),     # 10:00 AM
                'moto': df.loc[(df['id_start'] == id_start) & (df['id_end'] =
                'car': df.loc[(df['id_start'] == id_start) & (df['id_end'] ==
                'rv': df.loc[(df['id_start'] == id_start) & (df['id_end'] ==
                'bus': df.loc[(df['id_start'] == id_start) & (df['id_end'] ==
                'truck': df.loc[(df['id_start'] == id_start) & (df['id_end']
            })
            # Afternoon
            new_rows.append({
                'id_start': id_start,
                'id_end': id_end,
```

```
                'distance': distance,
                'start_day': day,
                'start_time': time(10, 0),    # 10:00 AM
                'end_day': day,
                'end_time': time(18, 0),       # 06:00 PM
                'moto': df.loc[(df['id_start'] == id_start) & (df['id_end'] =
                'car': df.loc[(df['id_start'] == id_start) & (df['id_end'] ==
                'rv': df.loc[(df['id_start'] == id_start) & (df['id_end'] ==
                'bus': df.loc[(df['id_start'] == id_start) & (df['id_end'] ==
                'truck': df.loc[(df['id_start'] == id_start) & (df['id_end']
            })
            # Evening
            new_rows.append({
                'id_start': id_start,
                'id_end': id_end,
                'distance': distance,
                'start_day': day,
                'start_time': time(18, 0),    # 06:00 PM
                'end_day': day,
                'end_time': time(23, 59),      # 11:59 PM
                'moto': df.loc[(df['id_start'] == id_start) & (df['id_end'] =
                'car': df.loc[(df['id_start'] == id_start) & (df['id_end'] ==
                'rv': df.loc[(df['id_start'] == id_start) & (df['id_end'] ==
                'bus': df.loc[(df['id_start'] == id_start) & (df['id_end'] ==
                'truck': df.loc[(df['id_start'] == id_start) & (df['id_end']
            })

        # Weekends (Saturday and Sunday)
        for day in days_of_week[5:]:  # Saturday and Sunday
            new_rows.append({
                'id_start': id_start,
                'id_end': id_end,
                'distance': distance,
                'start_day': day,
                'start_time': time(0, 0),    # 12:00 AM
                'end_day': day,
                'end_time': time(23, 59),      # 11:59 PM
                'moto': df.loc[(df['id_start'] == id_start) & (df['id_end'] =
                'car': df.loc[(df['id_start'] == id_start) & (df['id_end'] ==
                'rv': df.loc[(df['id_start'] == id_start) & (df['id_end'] ==
                'bus': df.loc[(df['id_start'] == id_start) & (df['id_end'] ==
                'truck': df.loc[(df['id_start'] == id_start) & (df['id_end']
            })

    # Create a new DataFrame from the new rows
    return pd.DataFrame(new_rows)

# Sample DataFrame Creation
def create_sample_toll_rates_df() -> pd.DataFrame:
    """
    Create a sample DataFrame to simulate the toll rates.

    Returns:
        pd.DataFrame: Sample DataFrame with id_start, id_end, distance, and t
    """
    # Sample data including the new entries
    data = {
```

```
        'id_start': [1001400, 1001402, 1001404, 1001408, 1001400, 1001408],
        'id_end': [1001402, 1001404, 1001406, 1001410, 1001402, 1001410],
        'distance': [10.0, 20.0, 30.0, 11.1, 9.7, 11.1],
        'moto': [9.7, 20.2, 16.0, 12.5, 9.7, 12.5],
        'car': [12.0, 22.0, 18.0, 15.0, 12.0, 15.0],
        'rv': [15.0, 25.0, 20.0, 17.0, 15.0, 17.0],
        'bus': [18.0, 28.0, 22.0, 19.0, 18.0, 19.0],
        'truck': [25.0, 35.0, 30.0, 27.0, 25.0, 27.0]
    }

    return pd.DataFrame(data)

# Main Execution
# Create a sample DataFrame
toll_rates_df = create_sample_toll_rates_df()

# Calculate time-based toll rates
result_df = calculate_time_based_toll_rates(toll_rates_df)

# Display the result DataFrame
print(result_df)
```

```
     id_start     id_end  distance start_day start_time   end_day  end_time
\
0   1001400.0  1001402.0      10.0    Monday   00:00:00    Monday  10:00:00
1   1001400.0  1001402.0      10.0    Monday   10:00:00    Monday  18:00:00
2   1001400.0  1001402.0      10.0    Monday   18:00:00    Monday  23:59:00
3   1001400.0  1001402.0      10.0   Tuesday   00:00:00   Tuesday  10:00:00
4   1001400.0  1001402.0      10.0   Tuesday   10:00:00   Tuesday  18:00:00
..        ...        ...       ...       ...        ...       ...       ...
80  1001400.0  1001402.0       9.7    Friday   00:00:00    Friday  10:00:00
81  1001400.0  1001402.0       9.7    Friday   10:00:00    Friday  18:00:00
82  1001400.0  1001402.0       9.7    Friday   18:00:00    Friday  23:59:00
83  1001400.0  1001402.0       9.7  Saturday   00:00:00  Saturday  23:59:00
84  1001400.0  1001402.0       9.7    Sunday   00:00:00    Sunday  23:59:00

     moto   car    rv   bus  truck
0    7.76   9.6  12.0  14.4   20.0
1   11.64  14.4  18.0  21.6   30.0
2    7.76   9.6  12.0  14.4   20.0
3    7.76   9.6  12.0  14.4   20.0
4   11.64  14.4  18.0  21.6   30.0
..    ...   ...   ...   ...    ...
80   7.76   9.6  12.0  14.4   20.0
81  11.64  14.4  18.0  21.6   30.0
82   7.76   9.6  12.0  14.4   20.0
83   6.79   8.4  10.5  12.6   17.5
84   6.79   8.4  10.5  12.6   17.5

[85 rows x 12 columns]
```

In [ ]: