In [1]:

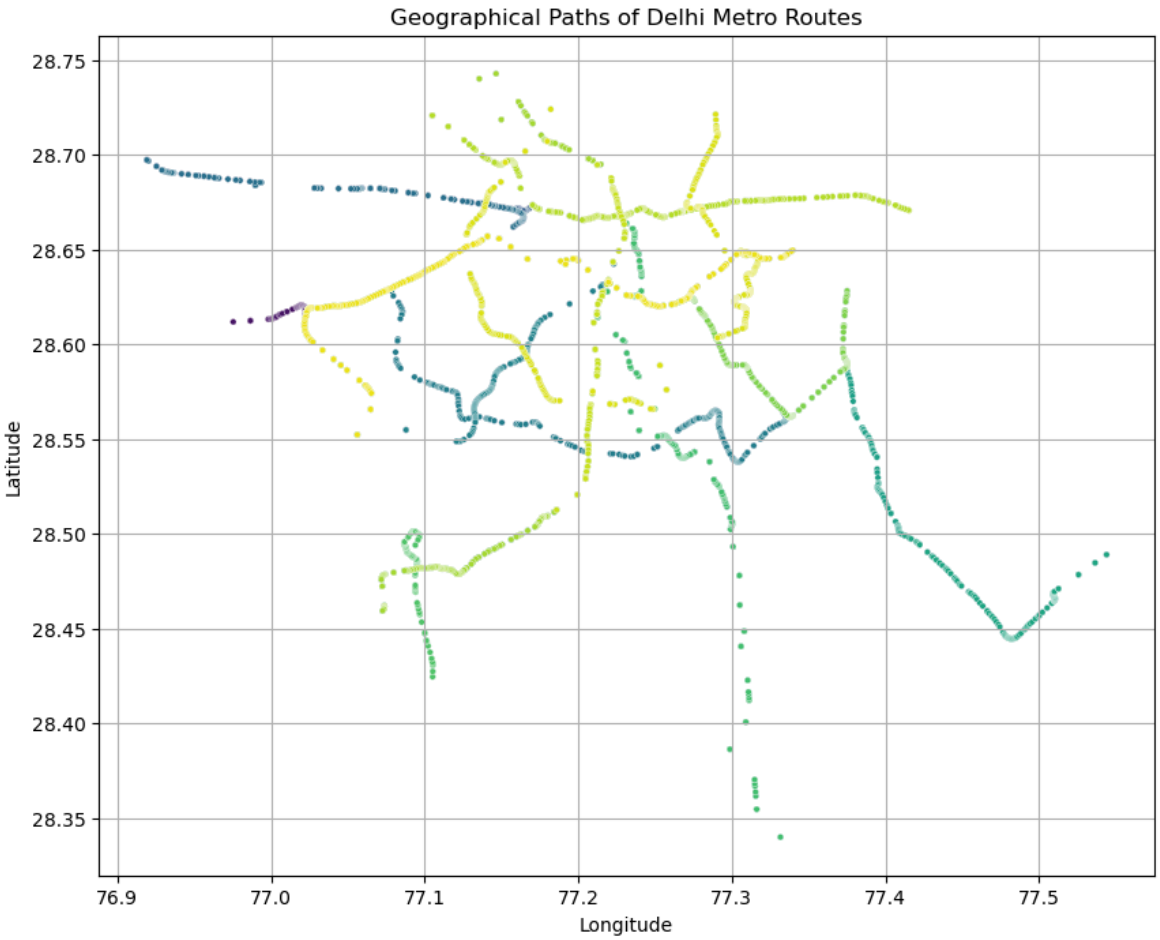
```python
import pandas as pd

# Define the directory path
directory = r'C:\Users\Sharon\Desktop\Docs\DMRC'

# Load all the data files
agency = pd.read_csv(f'{directory}\\agency.txt')
calendar = pd.read_csv(f'{directory}\\calendar.txt')
routes = pd.read_csv(f'{directory}\\routes.txt')
shapes = pd.read_csv(f'{directory}\\shapes.txt')
stop_times = pd.read_csv(f'{directory}\\stop_times.txt')
stops = pd.read_csv(f'{directory}\\stops.txt')
trips = pd.read_csv(f'{directory}\\trips.txt')

# Show the first few rows and the structure of each dataframe
data_overviews = {
    "agency": agency.head(),
    "calendar": calendar.head(),
    "routes": routes.head(),
    "shapes": shapes.head(),
    "stop_times": stop_times.head(),
    "stops": stops.head(),
    "trips": trips.head()
}

# Print an overview of the data
for key, value in data_overviews.items():
    print(f"Data Overview: {key}")
    print(value)
    print("\n")
```

```
Data Overview: agency
  agency_id                  agency_name                       agency_url  \
0      DMRC  Delhi Metro Rail Corporation  http://www.delhimetrorail.com/

  agency_timezone  agency_lang  agency_phone  agency_fare_url  agency_email
0    Asia/Kolkata          NaN           NaN              NaN           NaN


Data Overview: calendar
  service_id  monday  tuesday  wednesday  thursday  friday  saturday  sunday  \
0    weekday       1        1          1         1       1         0       0
1   saturday       0        0          0         0       0         1       0
2     sunday       0        0          0         0       0         0       1

   start_date  end_date
0    20190101  20251231
1    20190101  20251231
2    20190101  20251231


Data Overview: routes
   route_id  agency_id route_short_name  \
0        33        NaN           R_SP_R
1        31        NaN           G_DD_R
2        29        NaN           P_MS_R
3        12        NaN             M_JB
4        11        NaN             P_MS

                                   route_long_name  route_desc  route_type  \
0  RAPID_Phase 3 (Rapid Metro) to Sector 55-56 (R...         NaN           1
1                    GRAY_Dhansa Bus Stand to Dwarka         NaN           1
2                     PINK_Shiv Vihar to Majlis Park         NaN           1
3        MAGENTA_Janak Puri West to Botanical Garden         NaN           1
4                     PINK_Majlis Park to Shiv Vihar         NaN           1

   route_url  route_color  route_text_color  route_sort_order  \
0        NaN          NaN               NaN               NaN
1        NaN          NaN               NaN               NaN
2        NaN          NaN               NaN               NaN
3        NaN          NaN               NaN               NaN
4        NaN          NaN               NaN               NaN

   continuous_pickup  continuous_drop_off
0                NaN                  NaN
1                NaN                  NaN
2                NaN                  NaN
3                NaN                  NaN
4                NaN                  NaN


Data Overview: shapes
  shape_id  shape_pt_lat  shape_pt_lon  shape_pt_sequence  shape_dist_traveled
0  shp_1_2     28.615887     77.022461                  1                0.000
1  shp_1_2     28.616341     77.022499                  2               50.510
2  shp_1_2     28.617985     77.022453                  3              233.586
3  shp_1_2     28.618252     77.022453                  4              263.487
4  shp_1_2     28.618425     77.022438                  5              282.857


Data Overview: stop_times
```

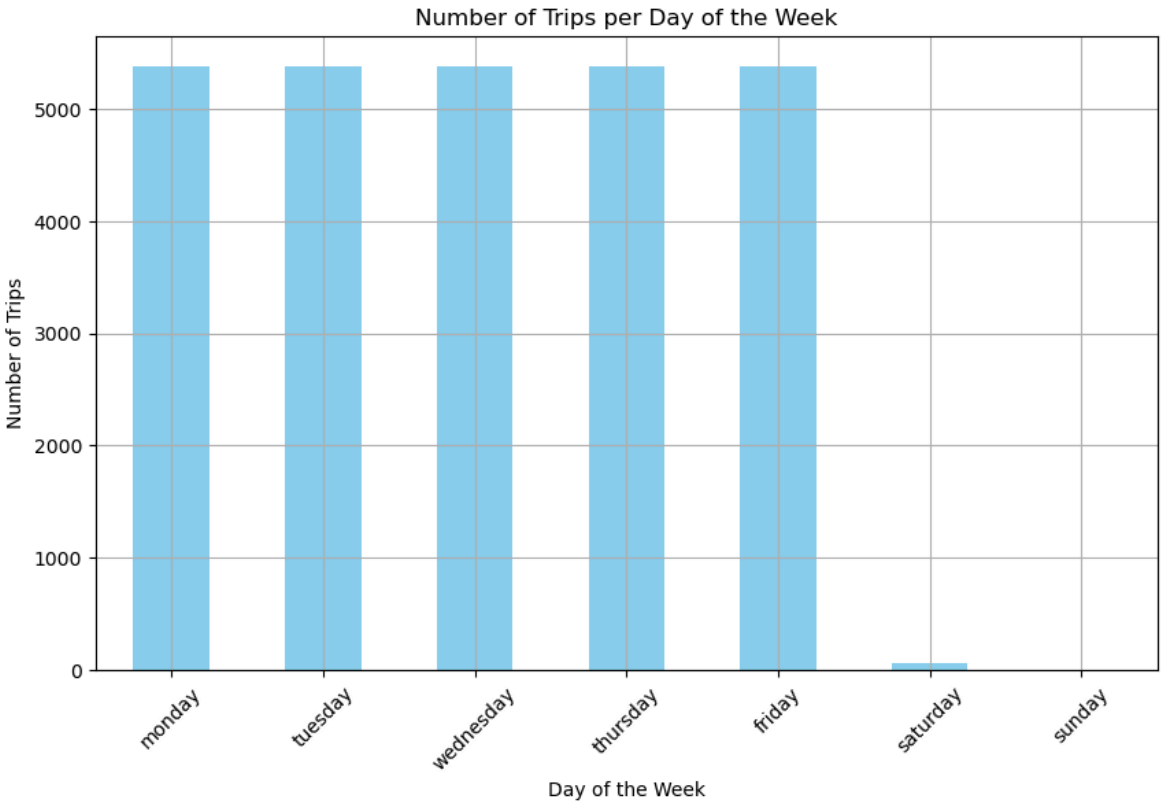
```
   trip_id arrival_time departure_time  stop_id  stop_sequence  stop_headsign  \
0        0     05:28:08       05:28:28       21              0            NaN
1        0     05:30:58       05:31:18       20              1            NaN
2        0     05:33:28       05:33:48       19              2            NaN
3        0     05:35:33       05:35:53       18              3            NaN
4        0     05:37:53       05:38:13       17              4            NaN

   pickup_type  drop_off_type  shape_dist_traveled  timepoint  \
0            0              0                0.000          1
1            0              0             1202.405          1
2            0              0             2480.750          1
3            0              0             3314.936          1
4            0              0             4300.216          1

   continuous_pickup  continuous_drop_off
0                NaN                  NaN
1                NaN                  NaN
2                NaN                  NaN
3                NaN                  NaN
4                NaN                  NaN


Data Overview: stops
   stop_id  stop_code       stop_name  stop_desc   stop_lat   stop_lon
0        1        NaN  Dilshad Garden        NaN  28.675991  77.321495
1        2        NaN         Jhilmil        NaN  28.675648  77.312393
2        3        NaN  Mansrover park        NaN  28.675352  77.301178
3        4        NaN        Shahdara        NaN  28.673531  77.287270
4        5        NaN         Welcome        NaN  28.671986  77.277931


Data Overview: trips
   route_id service_id  trip_id  trip_headsign  trip_short_name  direction_id  \
0         0    weekday        0            NaN              NaN           NaN
1         0    weekday        1            NaN              NaN           NaN
2         0    weekday       10            NaN              NaN           NaN
3         0    weekday      100            NaN              NaN           NaN
4         2    weekday     1000            NaN              NaN           NaN

   block_id  shape_id  wheelchair_accessible  bikes_allowed
0       NaN  shp_1_30                      0              0
1       NaN  shp_1_30                      0              0
2       NaN  shp_1_30                      0              0
3       NaN  shp_1_30                      0              0
4       NaN  shp_1_13                      0              0
```

In [2]:

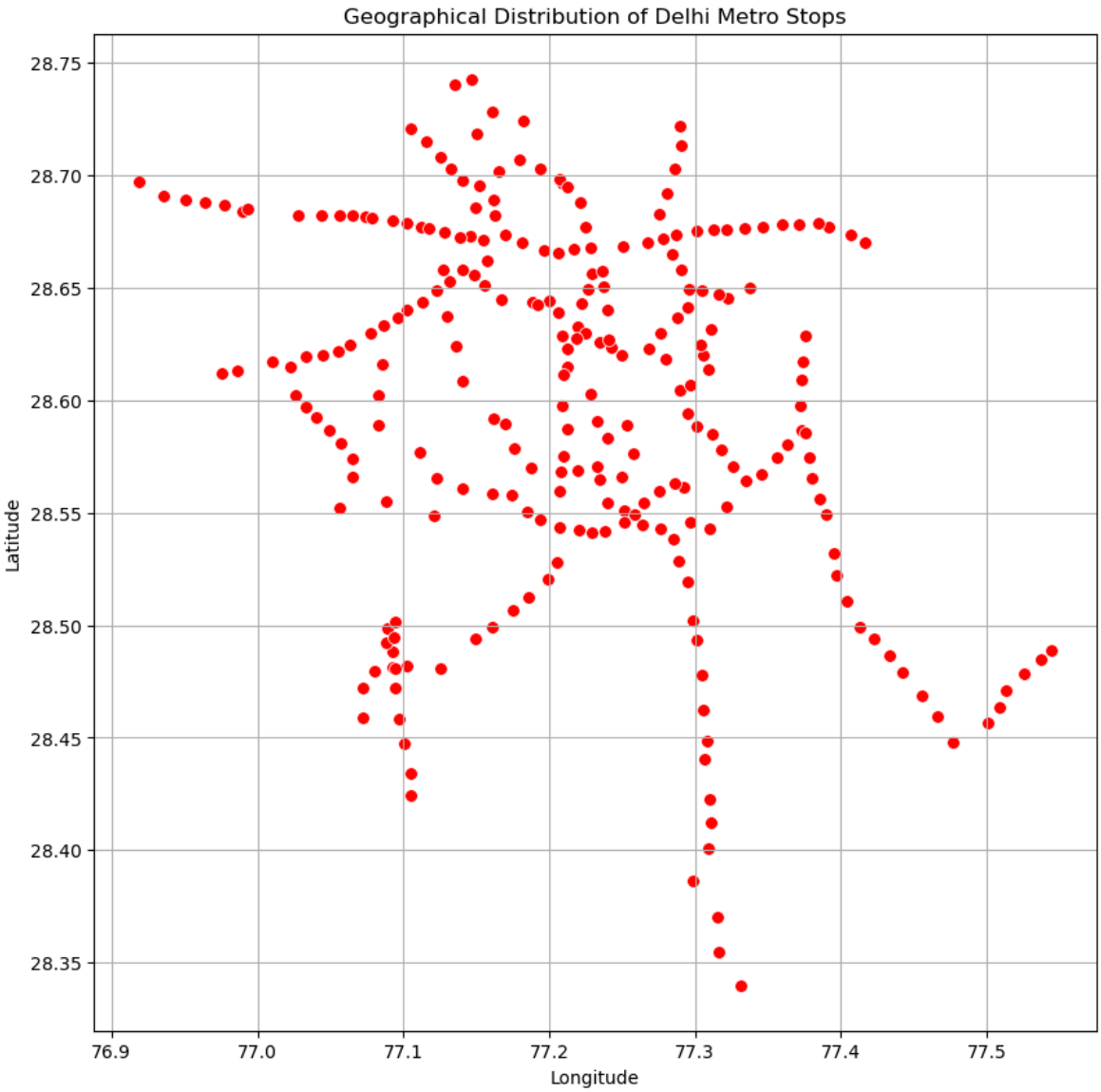
```python
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 8))
sns.scatterplot(x='shape_pt_lon', y='shape_pt_lat', hue='shape_id', data=shapes,
plt.title('Geographical Paths of Delhi Metro Routes')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.grid(True)
plt.show()
```



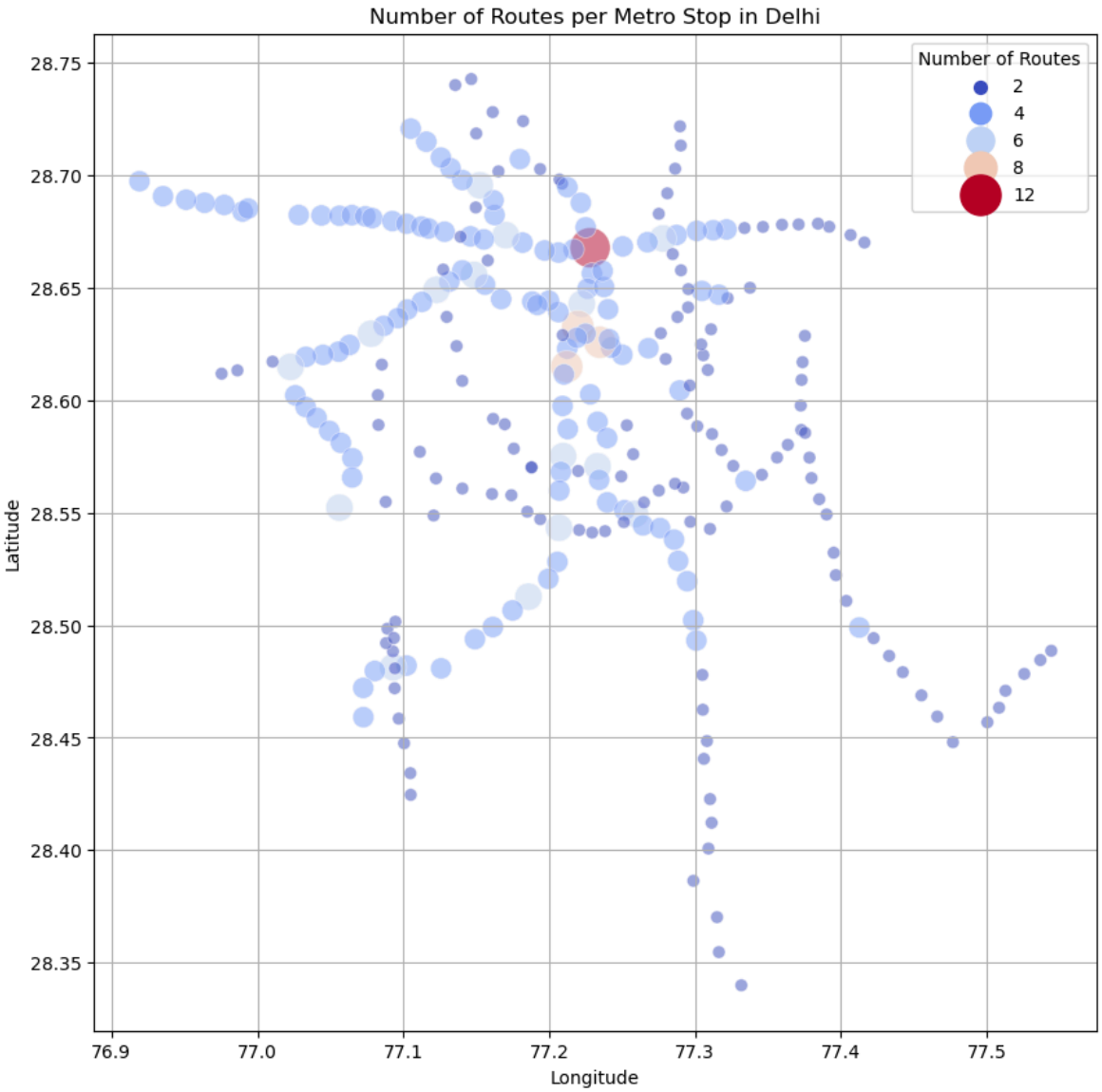
In [3]:
```
# merge trips with calendar to include the day of operation information
trips_calendar = pd.merge(trips, calendar, on='service_id', how='left')

# count the number of trips per day of the week
trip_counts = trips_calendar[['monday', 'tuesday', 'wednesday', 'thursday', 'fri

# Plotting
plt.figure(figsize=(10, 6))
trip_counts.plot(kind='bar', color='skyblue')
plt.title('Number of Trips per Day of the Week')
plt.xlabel('Day of the Week')
plt.ylabel('Number of Trips')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



In [4]:
```
# plotting the locations of the stops
plt.figure(figsize=(10, 10))
sns.scatterplot(x='stop_lon', y='stop_lat', data=stops, color='red', s=50, marke
plt.title('Geographical Distribution of Delhi Metro Stops')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.grid(True)
plt.show()
```



In [5]:
```
# merge stops with stop_times to link each stop with trips, and then merge with
stops_with_routes = pd.merge(pd.merge(stop_times, trips, on='trip_id'), routes,

# count how many unique routes pass through each stop
stop_route_counts = stops_with_routes.groupby('stop_id')['route_id'].nunique().r
stop_route_counts = stop_route_counts.rename(columns={'route_id': 'number_of_rou

# merge this with stops to get the names and location for plotting
stop_route_counts = pd.merge(stop_route_counts, stops, on='stop_id')

# plot the number of routes per stop
plt.figure(figsize=(10, 10))
sns.scatterplot(x='stop_lon', y='stop_lat', size='number_of_routes', hue='number
                sizes=(50, 500), alpha=0.5, palette='coolwarm', data=stop_route_
plt.title('Number of Routes per Metro Stop in Delhi')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.legend(title='Number of Routes')
plt.grid(True)
plt.show()
```

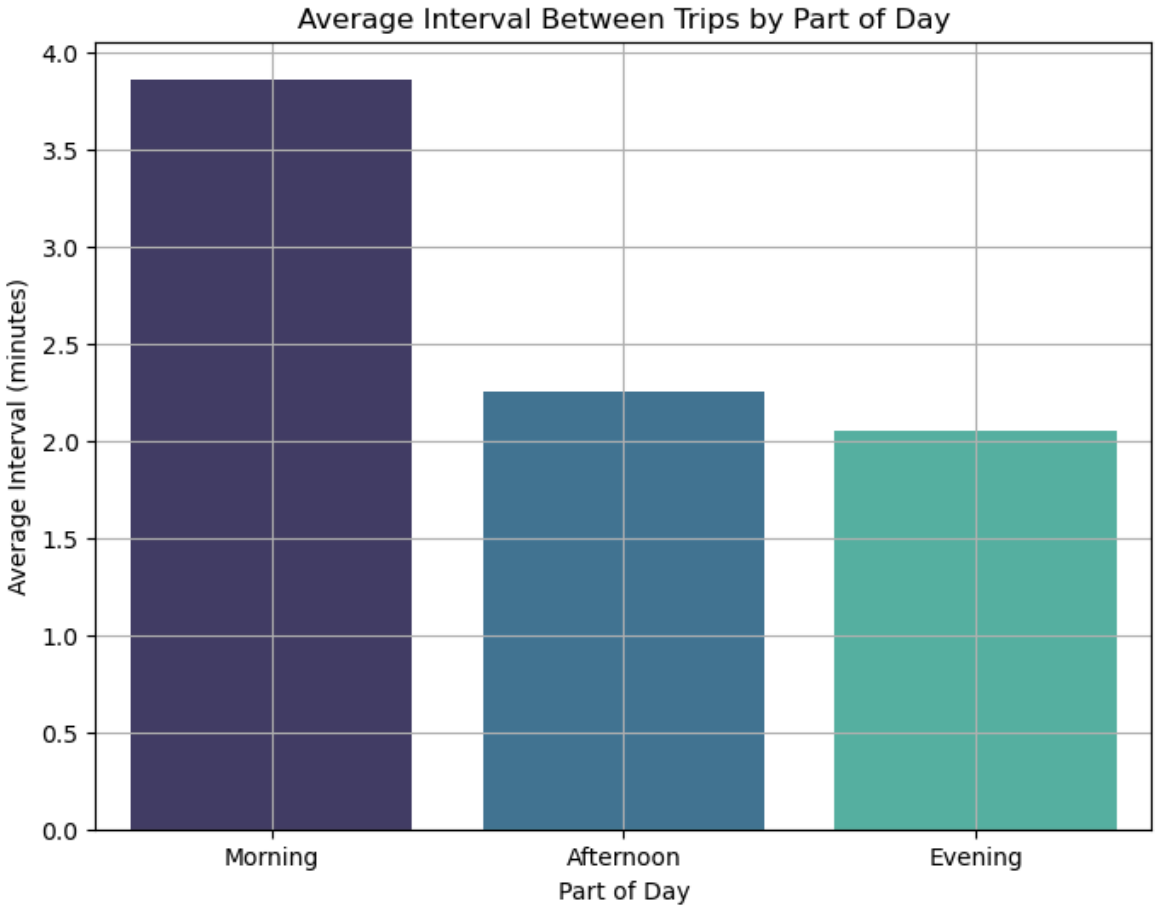


In [6]:

```python
# converting stop_times 'arrival_time' from string to datetime.time for easier m
import datetime as dt

# function to convert time string to datetime.time
def convert_to_time(time_str):
    try:
        return dt.datetime.strptime(time_str, '%H:%M:%S').time()
    except ValueError:
        # Handle cases where the hour might be greater than 23 (e.g., 24:00:00 o
        hour, minute, second = map(int, time_str.split(':'))
        return dt.time(hour % 24, minute, second)

stop_times['arrival_time_dt'] = stop_times['arrival_time'].apply(convert_to_time

# calculate the difference in arrival times for subsequent trips at each stop
stop_times_sorted = stop_times.sort_values(by=['stop_id', 'arrival_time_dt'])
stop_times_sorted['next_arrival_time'] = stop_times_sorted.groupby('stop_id')['a

# function to calculate the difference in minutes between two times
def time_difference(time1, time2):
    if pd.isna(time1) or pd.isna(time2):
        return None
    full_date_time1 = dt.datetime.combine(dt.date.today(), time1)
    full_date_time2 = dt.datetime.combine(dt.date.today(), time2)
    return (full_date_time2 - full_date_time1).seconds / 60
```

```
stop_times_sorted['interval_minutes'] = stop_times_sorted.apply(lambda row: time

# drop NaN values from intervals (last trip of the day)
stop_times_intervals = stop_times_sorted.dropna(subset=['interval_minutes'])

# average intervals by time of day (morning, afternoon, evening)
def part_of_day(time):
    if time < dt.time(12, 0):
        return 'Morning'
    elif time < dt.time(17, 0):
        return 'Afternoon'
    else:
        return 'Evening'

stop_times_intervals['part_of_day'] = stop_times_intervals['arrival_time_dt'].ap
average_intervals = stop_times_intervals.groupby('part_of_day')['interval_minute

plt.figure(figsize=(8, 6))
sns.barplot(x='part_of_day', y='interval_minutes', data=average_intervals, order
plt.title('Average Interval Between Trips by Part of Day')
plt.xlabel('Part of Day')
plt.ylabel('Average Interval (minutes)')
plt.grid(True)
plt.show()
```

```
C:\Users\Sharon\AppData\Local\Temp\ipykernel_13604\189162758.py:41: SettingWithCo
pyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stabl
e/user_guide/indexing.html#returning-a-view-versus-a-copy
  stop_times_intervals['part_of_day'] = stop_times_intervals['arrival_time_dt'].a
pply(part_of_day)
```

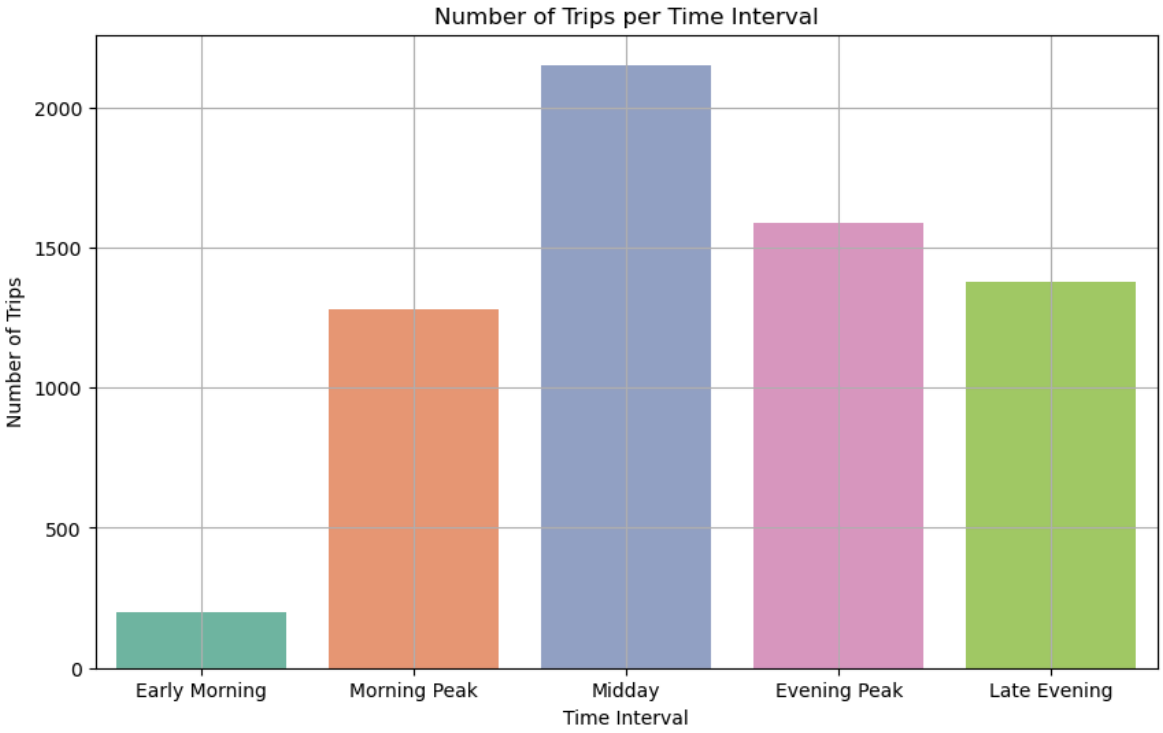


In [7]:

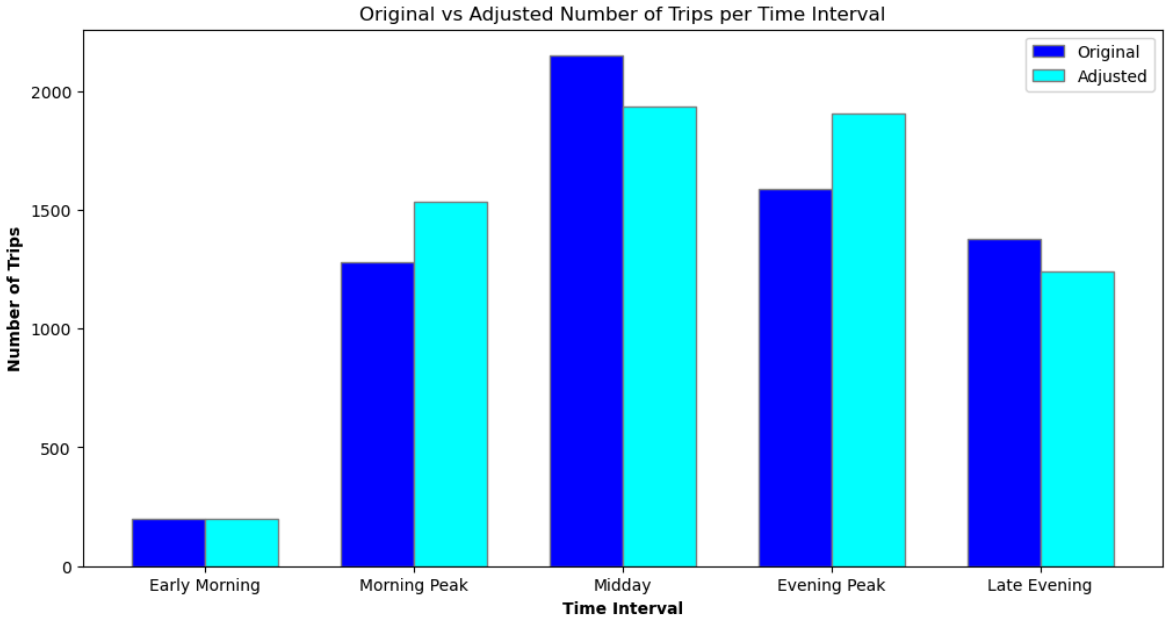
```
# define time intervals for classification
def classify_time_interval(time):
    if time < dt.time(6, 0):
        return 'Early Morning'
    elif time < dt.time(10, 0):
        return 'Morning Peak'
    elif time < dt.time(16, 0):
        return 'Midday'
    elif time < dt.time(20, 0):
        return 'Evening Peak'
    else:
        return 'Late Evening'

# apply time interval classification
stop_times['time_interval'] = stop_times['arrival_time_dt'].apply(classify_time_

# count the number of trips per time interval
trips_per_interval = stop_times.groupby('time_interval')['trip_id'].nunique().re
trips_per_interval = trips_per_interval.rename(columns={'trip_id': 'number_of_tr

# sorting the dataframe
ordered_intervals = ['Early Morning', 'Morning Peak', 'Midday', 'Evening Peak',
trips_per_interval['time_interval'] = pd.Categorical(trips_per_interval['time_in
trips_per_interval = trips_per_interval.sort_values('time_interval')

# plotting the number of trips per time interval
plt.figure(figsize=(10, 6))
sns.barplot(x='time_interval', y='number_of_trips', data=trips_per_interval, pal
plt.title('Number of Trips per Time Interval')
plt.xlabel('Time Interval')
plt.ylabel('Number of Trips')
```

```
plt.grid(True)
plt.show()
```

```
C:\Users\Sharon\anaconda3\Lib\site-packages\seaborn\categorical.py:641: FutureWar
ning: The default of observed=False is deprecated and will be changed to True in
a future version of pandas. Pass observed=False to retain current behavior or obs
erved=True to adopt the future default and silence this warning.
  grouped_vals = vals.groupby(grouper)
```



In [8]:
```
# adjusting frequencies based on hypothetical scenario
adjusted_trips_per_interval = trips_per_interval.copy()
adjustment_factors = {'Morning Peak': 1.20, 'Evening Peak': 1.20, 'Midday': 0.90

# apply the adjustments
adjusted_trips_per_interval['adjusted_number_of_trips'] = adjusted_trips_per_int
    lambda row: int(row['number_of_trips'] * adjustment_factors[row['time_interv

# plotting original vs adjusted number of trips per time interval
plt.figure(figsize=(12, 6))
bar_width = 0.35
r1 = range(len(adjusted_trips_per_interval))
r2 = [x + bar_width for x in r1]

plt.bar(r1, adjusted_trips_per_interval['number_of_trips'], color='blue', width=
plt.bar(r2, adjusted_trips_per_interval['adjusted_number_of_trips'], color='cyan

plt.xlabel('Time Interval', fontweight='bold')
plt.ylabel('Number of Trips', fontweight='bold')
plt.xticks([r + bar_width/2 for r in range(len(adjusted_trips_per_interval))], a
plt.title('Original vs Adjusted Number of Trips per Time Interval')
plt.legend()

plt.show()
```


Original vs Adjusted Number of Trips per Time Interval
Original
Adjusted
Number of Trips
0
500
1000
1500
2000
Early Morning
Morning Peak
Midday
Evening Peak
Late Evening
Time Interval

In [ ]: