metrooperations

June 8, 2024

[71]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

[43]:
```
agency = pd.read_csv('agency.txt')
calendar = pd.read_csv('calendar.txt')
routes = pd.read_csv('routes.txt')
shapes = pd.read_csv('shapes.txt')
stop_times = pd.read_csv('stop_times.txt')
stops = pd.read_csv('stops.txt')
trips = pd.read_csv('trips.txt')
```

[44]:
```
agency
```

[44]:
```
  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
```

[45]:
```
calendar
```

[45]:
```
  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
```

[46]:
```
routes.head()
```

[46]:
```
   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
```

[47]:
```
shapes.head()
```

[47]:

	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

[48]:
```
stop_times.head()
```

[48]:
```
   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
```

[49]:
```
stops.head()
```

[49]:
```
   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
```

[50]:
```
trips.head()
```

[50]:
```
   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
```

[51]:
```
columns = {
    'agency':agency.columns,
    'calendar':calendar.columns,
    'routes':routes.columns,
    'shapes':shapes.columns,
    'stop_times':stop_times.columns,
    'stops':stops.columns,
    'trips':trips.columns
}
```

[52]:
```
columns
```

[52]:
```
{'agency': Index(['agency_id', 'agency_name', 'agency_url', 'agency_timezone',
        'agency_lang', 'agency_phone', 'agency_fare_url', 'agency_email'],
       dtype='object'),
 'calendar': Index(['service_id', 'monday', 'tuesday', 'wednesday', 'thursday',
'friday',
        'saturday', 'sunday', 'start_date', 'end_date'],
       dtype='object'),
 'routes': Index(['route_id', 'agency_id', 'route_short_name',
'route_long_name',
        'route_desc', 'route_type', 'route_url', 'route_color',
        'route_text_color', 'route_sort_order', 'continuous_pickup',
        'continuous_drop_off'],
       dtype='object'),
 'shapes': Index(['shape_id', 'shape_pt_lat', 'shape_pt_lon',
'shape_pt_sequence',
        'shape_dist_traveled'],
       dtype='object'),
 'stop_times': Index(['trip_id', 'arrival_time', 'departure_time', 'stop_id',
'stop_sequence',
        'stop_headsign', 'pickup_type', 'drop_off_type', 'shape_dist_traveled',
        'timepoint', 'continuous_pickup', 'continuous_drop_off'],
       dtype='object'),
 'stops': Index(['stop_id', 'stop_code', 'stop_name', 'stop_desc', 'stop_lat',
        'stop_lon'],
       dtype='object'),
 'trips': Index(['route_id', 'service_id', 'trip_id', 'trip_headsign',
'trip_short_name',
        'direction_id', 'block_id', 'shape_id', 'wheelchair_accessible',
        'bikes_allowed'],
       dtype='object')}
```

[53]:
```
info = {
    'agency':agency.info(),
    'calendar':calendar.info(),
    'routes':routes.info(),
    'shapes':shapes.info(),
    'stop_times':stop_times.info(),
    'stops':stops.info(),
    'trips':trips.info()
}
```

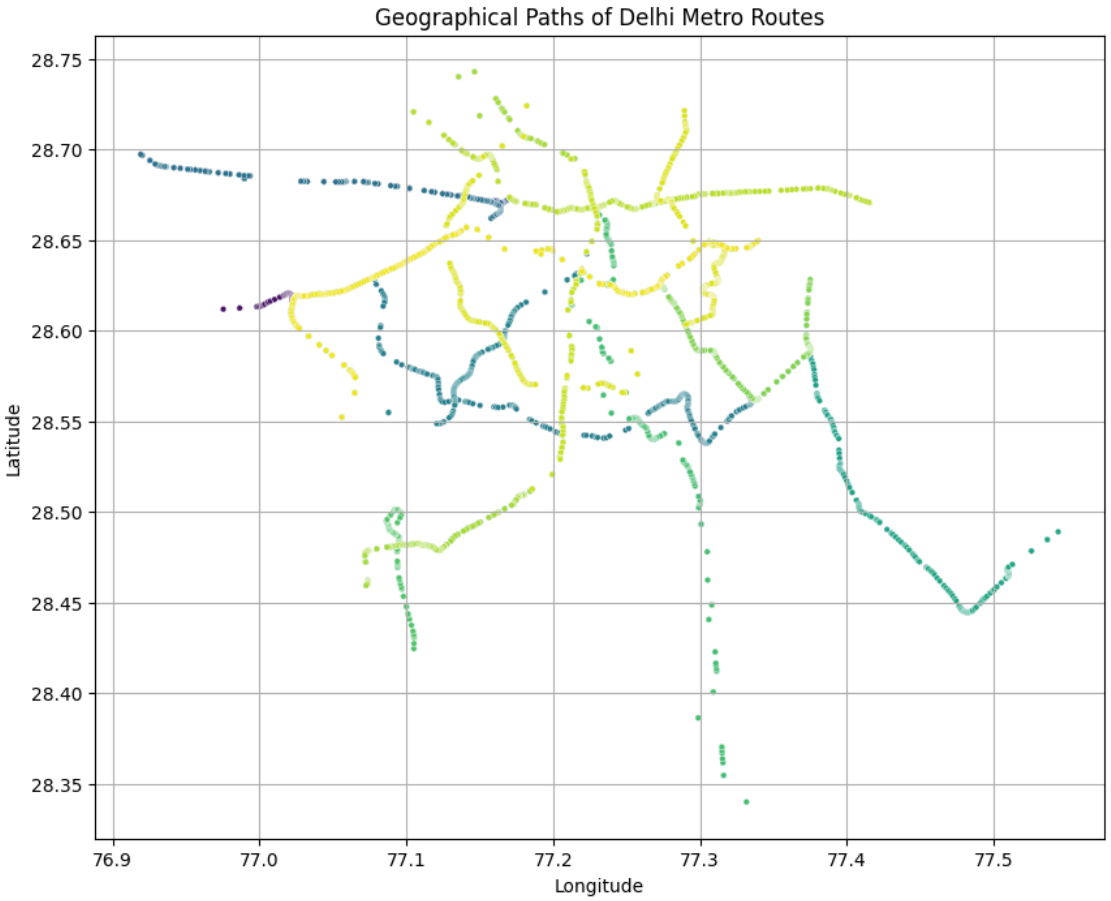
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1 entries, 0 to 0
Data columns (total 8 columns):
 #   Column           Non-Null Count  Dtype
```

```
---  ------           --------------  -----
 0   agency_id        1 non-null      object
 1   agency_name      1 non-null      object
 2   agency_url       1 non-null      object
 3   agency_timezone  1 non-null      object
 4   agency_lang      0 non-null      float64
 5   agency_phone     0 non-null      float64
 6   agency_fare_url  0 non-null      float64
 7   agency_email     0 non-null      float64
dtypes: float64(4), object(4)
memory usage: 192.0+ bytes
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3 entries, 0 to 2
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   service_id  3 non-null      object
 1   monday      3 non-null      int64
 2   tuesday     3 non-null      int64
 3   wednesday   3 non-null      int64
 4   thursday    3 non-null      int64
 5   friday      3 non-null      int64
 6   saturday    3 non-null      int64
 7   sunday      3 non-null      int64
 8   start_date  3 non-null      int64
 9   end_date    3 non-null      int64
dtypes: int64(9), object(1)
memory usage: 368.0+ bytes
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 12 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   route_id             36 non-null     int64
 1   agency_id            0 non-null      float64
 2   route_short_name     36 non-null     object
 3   route_long_name      36 non-null     object
 4   route_desc           0 non-null      float64
 5   route_type           36 non-null     int64
 6   route_url            0 non-null      float64
 7   route_color          0 non-null      float64
 8   route_text_color     0 non-null      float64
 9   route_sort_order     0 non-null      float64
 10  continuous_pickup    0 non-null      float64
 11  continuous_drop_off  0 non-null      float64
dtypes: float64(8), int64(2), object(2)
memory usage: 3.5+ KB
<class 'pandas.core.frame.DataFrame'>
```

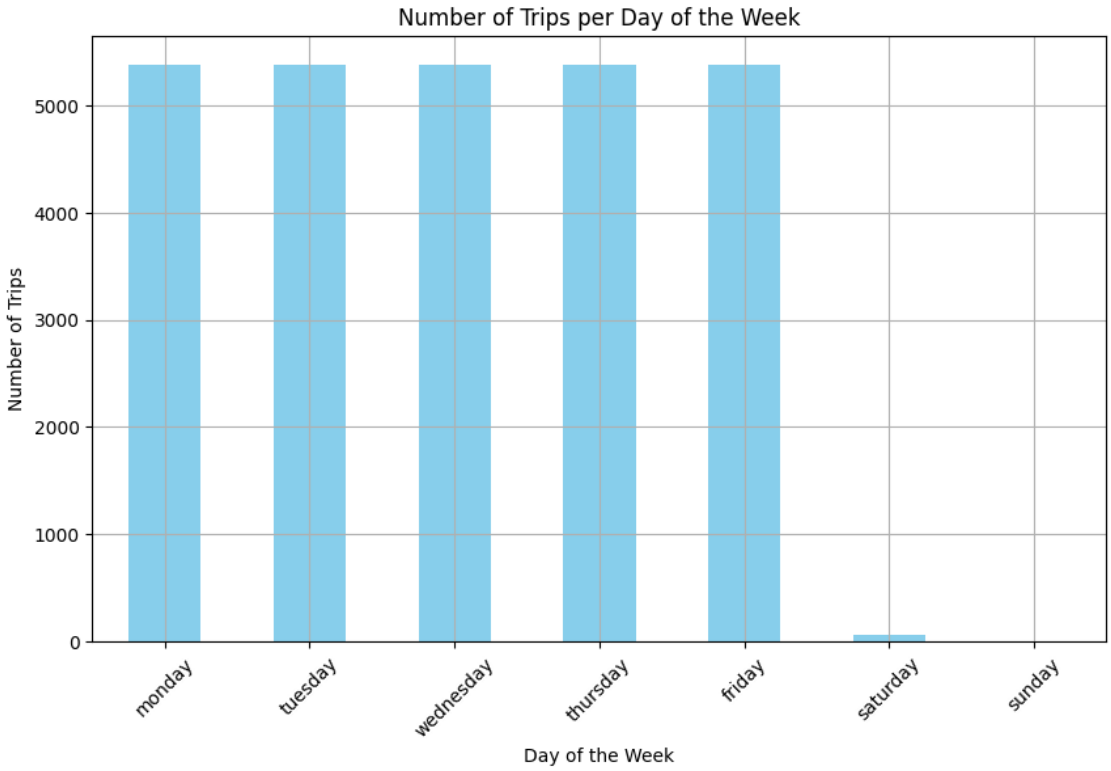
```
RangeIndex: 6643 entries, 0 to 6642
Data columns (total 5 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   shape_id             6643 non-null   object
 1   shape_pt_lat         6643 non-null   float64
 2   shape_pt_lon         6643 non-null   float64
 3   shape_pt_sequence    6643 non-null   int64
 4   shape_dist_traveled  6643 non-null   float64
dtypes: float64(3), int64(1), object(1)
memory usage: 259.6+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 128434 entries, 0 to 128433
Data columns (total 12 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   trip_id              128434 non-null  int64
 1   arrival_time         128434 non-null  object
 2   departure_time       128434 non-null  object
 3   stop_id              128434 non-null  int64
 4   stop_sequence        128434 non-null  int64
 5   stop_headsign        0 non-null       float64
 6   pickup_type          128434 non-null  int64
 7   drop_off_type        128434 non-null  int64
 8   shape_dist_traveled  128434 non-null  float64
 9   timepoint            128434 non-null  int64
 10  continuous_pickup    0 non-null       float64
 11  continuous_drop_off  0 non-null       float64
dtypes: float64(4), int64(6), object(2)
memory usage: 11.8+ MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 262 entries, 0 to 261
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   stop_id    262 non-null    int64
 1   stop_code  0 non-null      float64
 2   stop_name  262 non-null    object
 3   stop_desc  0 non-null      float64
 4   stop_lat   262 non-null    float64
 5   stop_lon   262 non-null    float64
dtypes: float64(4), int64(1), object(1)
memory usage: 12.4+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5438 entries, 0 to 5437
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
```

```
 0   route_id              5438 non-null   int64
 1   service_id            5438 non-null   object
 2   trip_id               5438 non-null   int64
 3   trip_headsign         0 non-null      float64
 4   trip_short_name       0 non-null      float64
 5   direction_id          0 non-null      float64
 6   block_id              0 non-null      float64
 7   shape_id              5438 non-null   object
 8   wheelchair_accessible 5438 non-null   int64
 9   bikes_allowed         5438 non-null   int64
dtypes: float64(4), int64(4), object(2)
memory usage: 425.0+ KB
```

[54]:
```
plt.figure(figsize=(10, 8))
sns.scatterplot(x='shape_pt_lon', y='shape_pt_lat', hue='shape_id',␣
 ↪data=shapes, palette='viridis', s=10, legend=None)
plt.title('Geographical Paths of Delhi Metro Routes')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.grid(True)
```



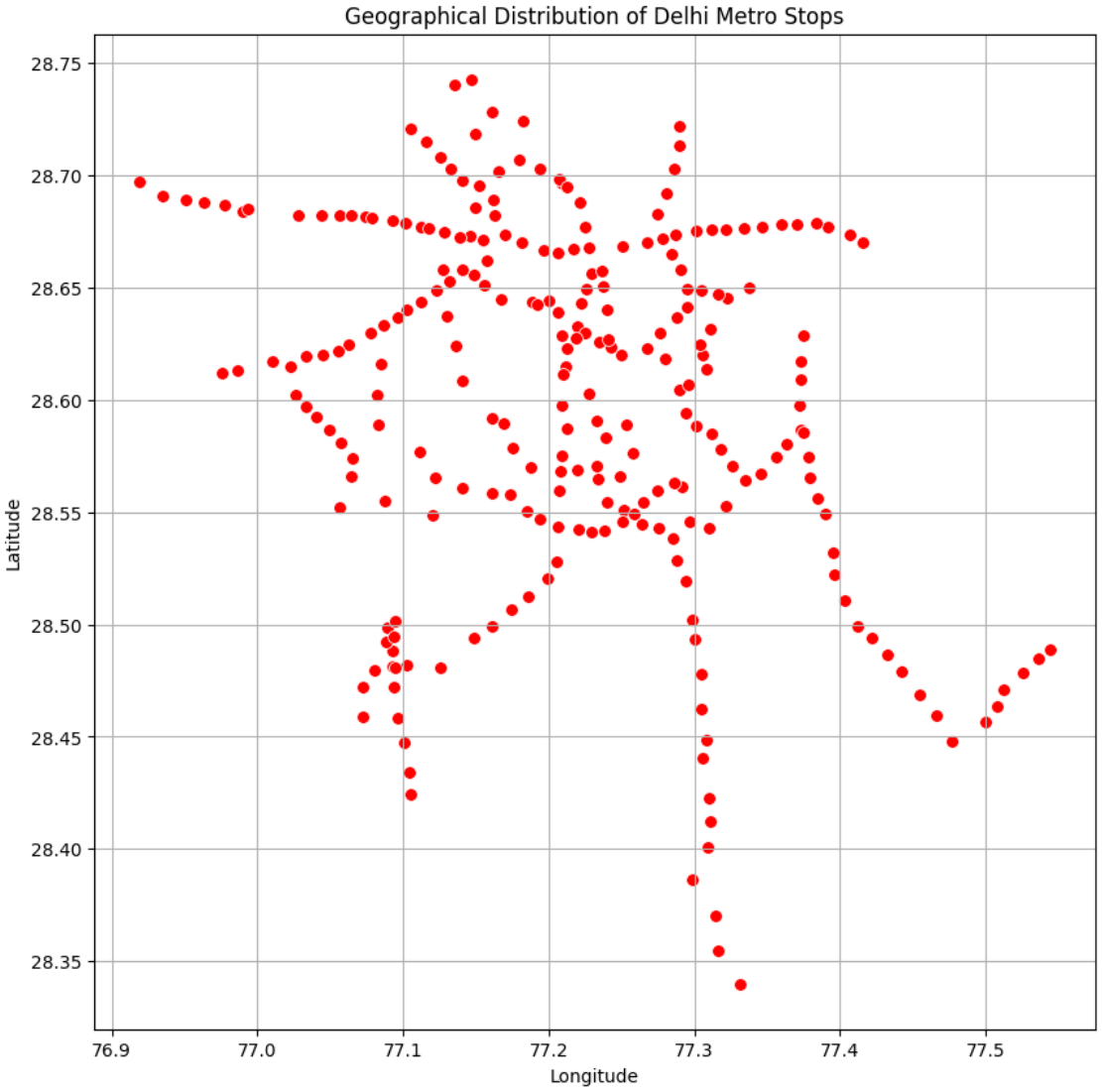
[55]:
```
trips_calendar = pd.merge(trips, calendar, on='service_id', how='left')
trip_counts = trips_calendar[['monday', 'tuesday', 'wednesday', 'thursday',␣
 ↪'friday', 'saturday', 'sunday']].sum()

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)
```



- Number of trips are consistent between monday to friday but it decreases to significant amount on saturday and sunday
- This indicates during week days demand is as its peak but decreases during off-days.

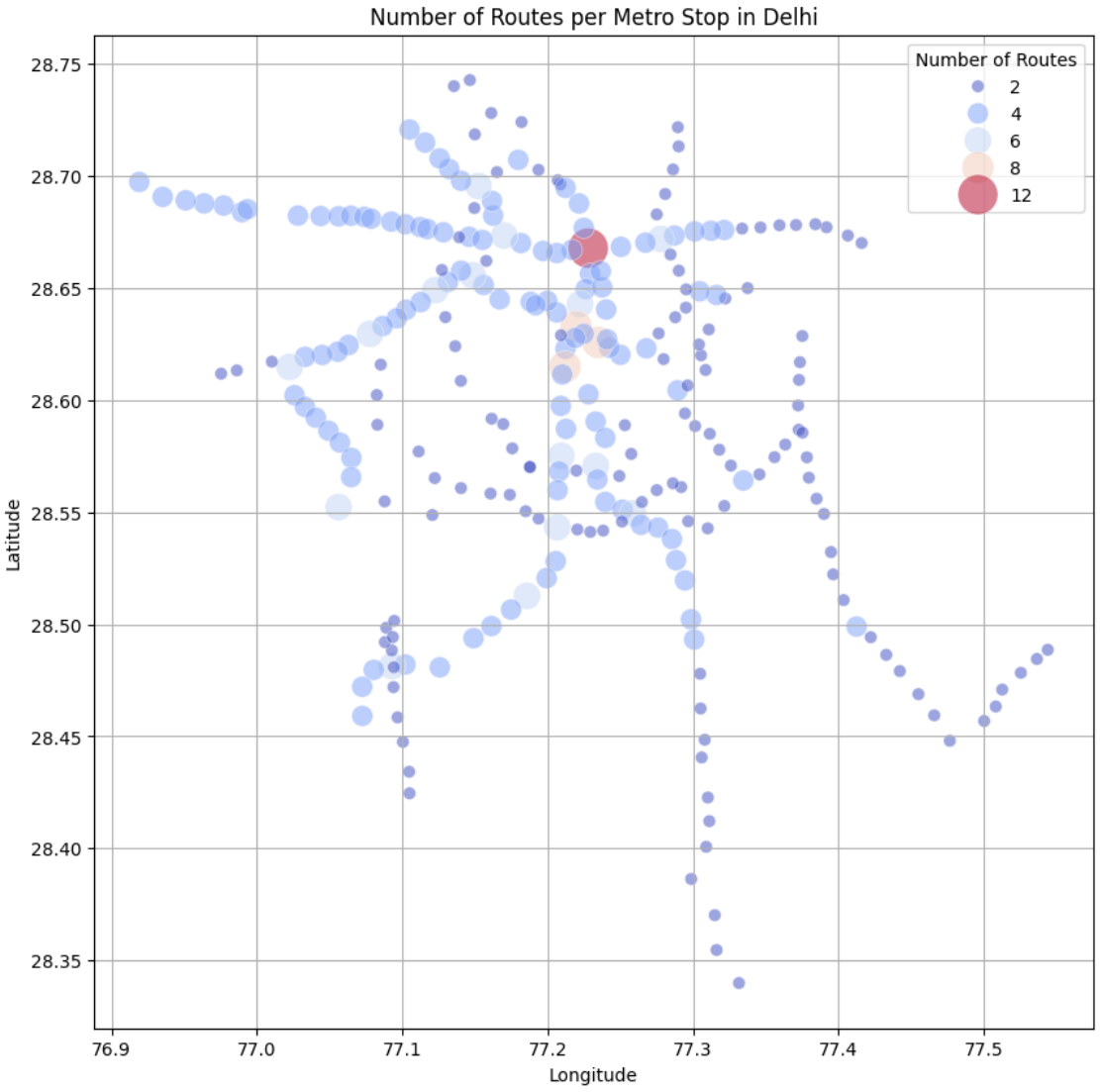
[56]:
```
plt.figure(figsize=(10, 10))
sns.scatterplot(x='stop_lon', y='stop_lat', data=stops, color='red', s=50,␣
 ↪marker='o')
plt.title('Geographical Distribution of Delhi Metro Stops')
plt.xlabel('Longitude')
```

```
plt.ylabel('Latitude')
plt.grid(True)
plt.show()
```



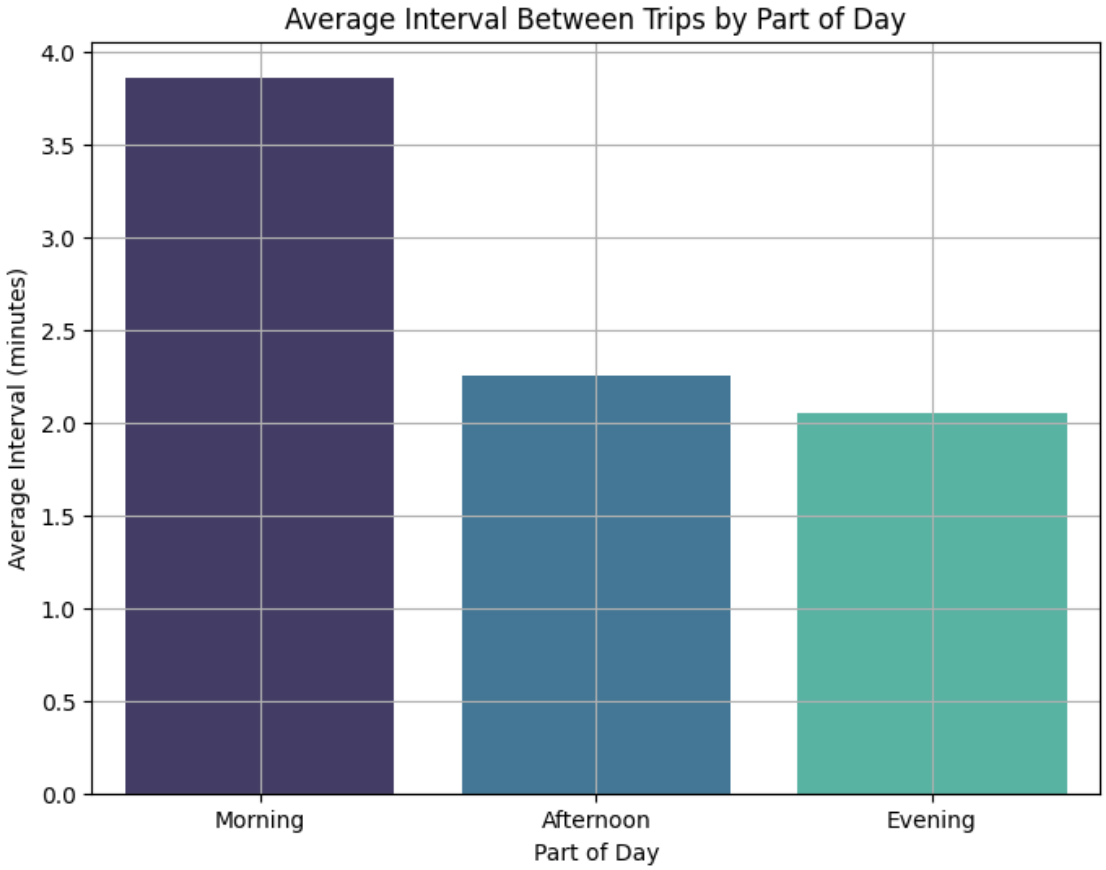
[57]:
```
stops_with_routes = pd.merge(pd.merge(stop_times, trips, on='trip_id'), routes,␣
↪on='route_id')

stop_route_counts = stops_with_routes.groupby('stop_id')['route_id'].nunique().
↪reset_index()
stop_route_counts = stop_route_counts.rename(columns={'route_id':␣
↪'number_of_routes'})

stop_route_counts = pd.merge(stop_route_counts, stops, on='stop_id')
```

```
plt.figure(figsize=(10, 10))
sns.scatterplot(x='stop_lon', y='stop_lat', size='number_of_routes',␣
 ↪hue='number_of_routes',
                sizes=(50, 500), alpha=0.5, palette='coolwarm',␣
 ↪data=stop_route_counts)
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()
```



[68]:
```
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␣
 ↪or 25:00:00)
        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')['arrival_time_dt'].shift(-1)

# 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_difference(row['arrival_time_dt'], row['next_arrival_time']), axis=1)

stop_times_intervals = stop_times_sorted.dropna(subset=['interval_minutes'])

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'].
 ↪apply(part_of_day)
average_intervals = stop_times_intervals.
 ↪groupby('part_of_day')['interval_minutes'].mean().reset_index()
```

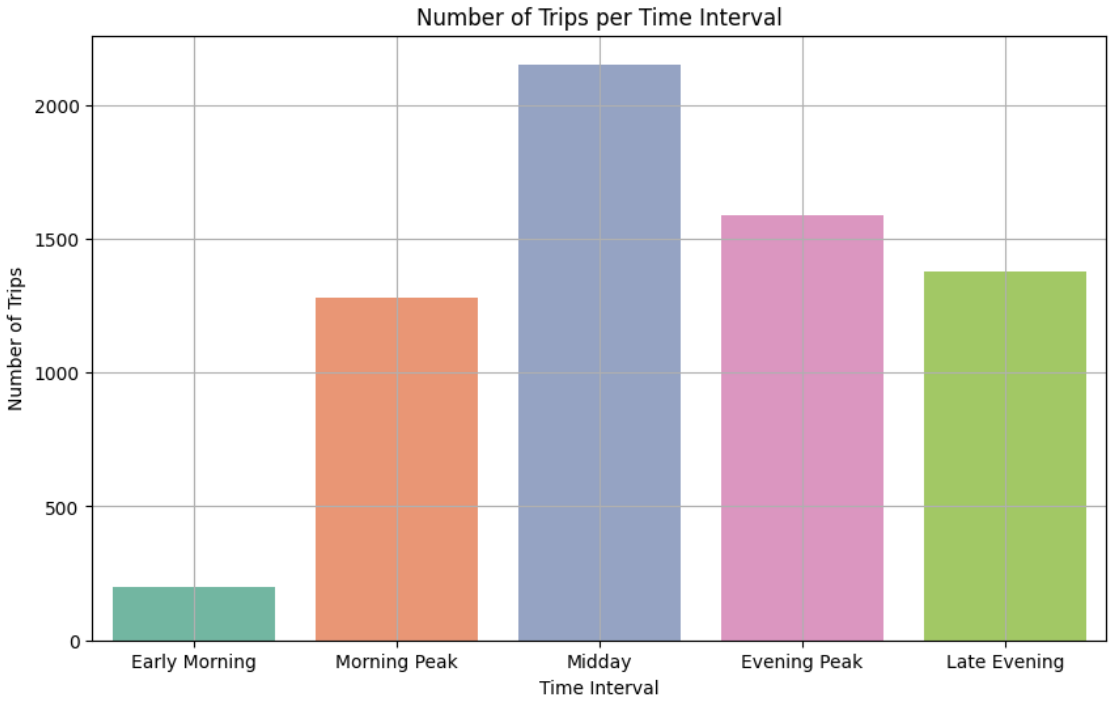
[69]:
```
plt.figure(figsize=(8, 6))
sns.barplot(x='part_of_day', y='interval_minutes', data=average_intervals,␣
↪order=['Morning', 'Afternoon', 'Evening'], palette='mako')
plt.title('Average Interval Between Trips by Part of Day')
plt.xlabel('Part of Day')
plt.ylabel('Average Interval (minutes)')
plt.grid(True)
plt.show()
```



[70]:
```
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_interval)

# count the number of trips per time interval
trips_per_interval = stop_times.groupby('time_interval')['trip_id'].nunique().
 ↪reset_index()
trips_per_interval = trips_per_interval.rename(columns={'trip_id':␣
 ↪'number_of_trips'})

# sorting the dataframe
ordered_intervals = ['Early Morning', 'Morning Peak', 'Midday', 'Evening Peak',␣
 ↪'Late Evening']
trips_per_interval['time_interval'] = pd.
 ↪Categorical(trips_per_interval['time_interval'],␣
 ↪categories=ordered_intervals, ordered=True)
trips_per_interval = trips_per_interval.sort_values('time_interval')
```

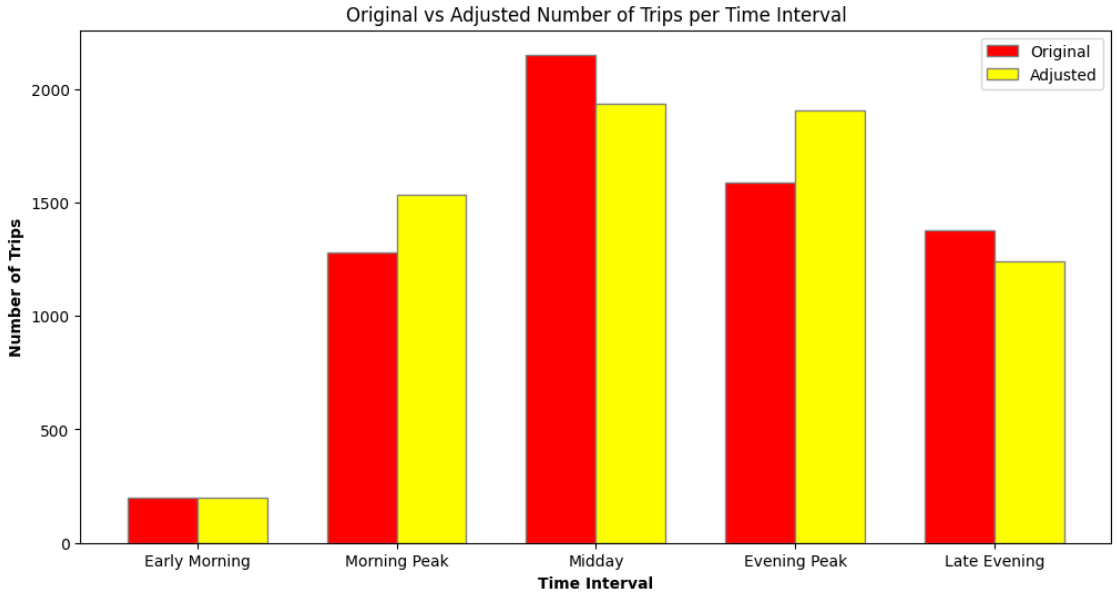
[67]:
```
plt.figure(figsize=(10, 6))
sns.barplot(x='time_interval', y='number_of_trips', data=trips_per_interval,␣
 ↪palette='Set2')
plt.title('Number of Trips per Time Interval')
plt.xlabel('Time Interval')
plt.ylabel('Number of Trips')
plt.grid(True)
plt.show()
```



[64]:
```
adjusted_trips_per_interval = trips_per_interval.copy()
adjustment_factors = {'Morning Peak': 1.20, 'Evening Peak': 1.20, 'Midday': 0.
 ↪90, 'Early Morning': 1.0, 'Late Evening': 0.90}

# apply the adjustments
adjusted_trips_per_interval['adjusted_number_of_trips'] =␣
 ↪adjusted_trips_per_interval.apply(
    lambda row: int(row['number_of_trips'] *␣
 ↪adjustment_factors[row['time_interval']]), axis=1)
```

[65]:
```
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='red',␣
 ↪width=bar_width, edgecolor='grey', label='Original')
plt.bar(r2, adjusted_trips_per_interval['adjusted_number_of_trips'],␣
 ↪color='yellow', width=bar_width, edgecolor='grey', label='Adjusted')

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))],␣
 ↪adjusted_trips_per_interval['time_interval'])
```

```
plt.title('Original vs Adjusted Number of Trips per Time Interval')
plt.legend()

plt.show()
```



[ ]: