In [51]:
```
import pandas as pd

# Load dataset
df = pd.read_csv('../data/jakarta_traffic_data.csv')

# Tampilkan 5 baris pertama
df.head()
```

Out[51]:

	Date	Location	Hour	Vehicle_Count	Average_Speed_kmh	Weather_Condition	Is_Weel
0	2024-01-01	Thamrin-Sudirman	7	1250.0	15.2	Sunny	I
1	2024-01-01	Thamrin-Sudirman	8	1890.0	12.5	Sunny	I
2	2024-01-01	Thamrin-Sudirman	9	1650.0	18.3	Sunny	I
3	2024-01-01	Thamrin-Sudirman	17	1780.0	14.1	Sunny	I
4	2024-01-01	Thamrin-Sudirman	18	2100.0	11.8	Sunny	I

In [52]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 288 entries, 0 to 287
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Date               288 non-null    object 
 1   Location           288 non-null    object 
 2   Hour               288 non-null    int64  
 3   Vehicle_Count      287 non-null    float64
 4   Average_Speed_kmh  287 non-null    float64
 5   Weather_Condition  287 non-null    object 
 6   Is_Weekend         287 non-null    object 
 7   Road_Type          288 non-null    object 
dtypes: float64(2), int64(1), object(5)
memory usage: 18.1+ KB
```

In [53]:
```
df.describe()
```

Out[53]:

	Hour	Vehicle_Count	Average_Speed_kmh
count	288.000000	287.000000	287.000000
mean	12.027778	1349.442509	21.431359
std	4.319060	412.906764	8.013873
min	7.000000	380.000000	8.400000
25%	8.000000	1050.000000	15.600000
50%	10.000000	1320.000000	19.800000
75%	17.000000	1650.000000	25.800000
max	22.000000	2450.000000	52.100000

Konversi kolom Date ke datetime (convertion date to datetime) Ini penting agar kamu bisa melakukan analisis berdasarkan hari/tanggal secara akurat.

In [54]:

```
df['Date'] = pd.to_datetime(df['Date'])
```

Konversi kolom Is_Weekend ke Boolean (is_weekend to boolean), Supaya nanti bisa difilter atau dianalisis dengan kondisi weekend lebih mudah.

In [55]:

```
df['Is_Weekend'] = df['Is_Weekend'].astype(bool)
```

📌 Cek & Tangani Missing Values, Untuk mengetahui kolom mana yang masih punya nilai kosong (NaN), dan berapa banyak.

In [56]:

```
df.isnull().sum()
```

Out[56]:

```
Date                 0
Location             0
Hour                 0
Vehicle_Count        1
Average_Speed_kmh    1
Weather_Condition    1
Is_Weekend           0
Road_Type            0
dtype: int64
```

🧼 Step 2: Data Cleaning and Preparation

🚩 **Tujuan:**

- Konversi kolom `Date` menjadi format datetime
- Isi missing values:
  - Angka → isi dengan **mean**
  - Kategori → isi dengan **modus**
- Tambahkan kolom baru:

  - `Day_of_Week`
  - `Time_Period`

In [57]:

```python
# Convert Date
df['Date'] = pd.to_datetime(df['Date'])

# Fill missing numerical values
df['Vehicle_Count'] = df['Vehicle_Count'].fillna(df['Vehicle_Count'].mean())
df['Average_Speed_kmh'] = df['Average_Speed_kmh'].fillna(df['Average_Speed_kmh'].me

# Fill missing categorical values
df['Weather_Condition'] = df['Weather_Condition'].fillna(df['Weather_Condition'].mo
df['Is_Weekend'] = df['Is_Weekend'].fillna(df['Is_Weekend'].mode()[0])
```

In [58]:

```python
#colum Day_of_Week (monday-sunday)
df['Day_of_Week'] = df['Date'].dt.day_name()

#make column Time_Period
def categorize_time(hour):
    if 7 <= hour <= 9:
        return "Morning Rush"
    elif 10 <= hour <= 15:
        return "Midday"
    elif 16 <= hour <= 19:
        return "Evening Rush"
    else:
        return "Night"

df['Time_Period'] = df['Hour'].apply(categorize_time)

#check new column
df[['Date', 'Hour', 'Day_of_Week', 'Time_Period']].head()
```

Out[58]:

	Date	Hour	Day_of_Week	Time_Period
0	2024-01-01	7	Monday	Morning Rush
1	2024-01-01	8	Monday	Morning Rush
2	2024-01-01	9	Monday	Morning Rush
3	2024-01-01	17	Monday	Evening Rush
4	2024-01-01	18	Monday	Evening Rush

✅Traffic Pattern Analysis

- 1. Peak Hours Analysis
  - a. Jam dengan kendaraan terbanyak (rata-rata tertinggi) >> highest hour (avarage high)
  - b. Jam dengan kecepatan paling rendah >>Lowest avarage speed
- 2. Location Comparison

    - a. Rata-rata kendaraan per lokasi & 3 lokasi terpadat (avarage per location & most location)
    - b. Lokasi dengan kecepatan rata-rata paling lambat (location which avarage speed slowest)

- 3. Weekend vs Weekday Analysis

In [59]:

```
#Hours with the most vehicles (highest average)
peak_vehicle_hour = df.groupby('Hour')['Vehicle_Count'].mean().idxmax()
print(f"Hour with the highest average vehicle count: {peak_vehicle_hour}:00")

#hours with the lowest speed
slowest_speed_hour = df.groupby('Hour')['Average_Speed_kmh'].mean().idxmin()
print(f"Hour with the lowest average speed: {slowest_speed_hour}:00")

#Average vehicles per location & 3 most crowded locations
avg_vehicle_by_location = df.groupby('Location')['Vehicle_Count'].mean().sort_value
print("\nTop 3 Most Congested Locations (by avg vehicle count):")
print(avg_vehicle_by_location.head(3))

#Locations with the slowest average speeds
slowest_location = df.groupby('Location')['Average_Speed_kmh'].mean().idxmin()
print(f"\nLocation with the lowest average speed: {slowest_location}")

#Weekend vs Weekday Analysis
weekend_group = df.groupby('Is_Weekend')

print("\nAverage Vehicle Count (Weekend vs Weekday):")
print(weekend_group['Vehicle_Count'].mean())

print("\nAverage Speed (Weekend vs Weekday):")
print(weekend_group['Average_Speed_kmh'].mean())
```

```
Hour with the highest average vehicle count: 18:00
Hour with the lowest average speed: 18:00

Top 3 Most Congested Locations (by avg vehicle count):
Location
Thamrin-Sudirman    1663.325590
Senayan_Circle      1452.777778
Gatot_Subroto       1283.194444
Name: Vehicle_Count, dtype: float64

Location with the lowest average speed: Thamrin-Sudirman

Average Vehicle Count (Weekend vs Weekday):
Is_Weekend
False    1456.320370
True      982.769231
Name: Vehicle_Count, dtype: float64

Average Speed (Weekend vs Weekday):
Is_Weekend
False    18.778168
True     30.533846
Name: Average_Speed_kmh, dtype: float64
```

✅ Analisis Dampak Cuaca terhadap Lalu Lintas (Weather Impact Analysis on Traffic)

- 1. Rata-rata jumlah kendaraan berdasarkan cuaca (Average number of vehicles based on weather)
- 2. Rata-rata kecepatan berdasarkan cuaca (Average speed based on weather)
- 3. Cuaca paling parah → kecepatan paling rendah (Worst weather → lowest speed)
- 4. Persentase perbedaan kecepatan antara cerah dan hujan (Percentage difference in speed between sunny and rainy)

In [60]:

```python
vehicle_by_weather = df.groupby('Weather_Condition')['Vehicle_Count'].mean()
print("Average Vehicle Count by Weather Condition:")
print(vehicle_by_weather)

speed_by_weather = df.groupby('Weather_Condition')['Average_Speed_kmh'].mean()
print("\nAverage Speed by Weather Condition:")
print(speed_by_weather)

worst_weather = speed_by_weather.idxmin()
print(f"\nWeather condition with the lowest average speed (worst for traffic): {wor

sunny_speed = speed_by_weather.get('Sunny', None)
rainy_speed = speed_by_weather.get('Rainy', None)

if sunny_speed and rainy_speed:
    diff_percent = ((sunny_speed - rainy_speed) / sunny_speed) * 100
    print(f"\nSpeed drops by {diff_percent:.2f}% on Rainy days compared to Sunny da
else:
    print("\nSpeed comparison between Sunny and Rainy days not possible (data missi
Average Vehicle Count by Weather Condition:
Weather_Condition
Cloudy     1314.285714
Rainy      1500.789474
Sunny      1284.292167
Name: Vehicle_Count, dtype: float64

Average Speed by Weather Condition:
Weather_Condition
Cloudy     21.862745
Rainy      18.244737
Sunny      22.979259
Name: Average_Speed_kmh, dtype: float64

Weather condition with the lowest average speed (worst for traffic): Rainy

Speed drops by 20.60% on Rainy days compared to Sunny days.
```

✅ Analisis Performa Berdasarkan Tipe Jalan (Performance Analysis Based on Road Type)

- Tujuannya:
  - Bandingkan rata-rata volume kendaraan per tipe jalan
  - Bandingkan rata-rata kecepatan per tipe jalan
  - Tentukan jalan paling padat dan jalan paling lancar
- Objectives:
  - Compare average vehicle volume per road type
  - Compare average speed per road type
  - Determine the most congested and smoothest roads

In [61]:

```
#Average number of vehicles per road type
vehicle_by_road = df.groupby('Road_Type')['Vehicle_Count'].mean().sort_values(ascen
print("Average Vehicle Count by Road Type:")
print(vehicle_by_road)

#Average speed per road type
speed_by_road = df.groupby('Road_Type')['Average_Speed_kmh'].mean().sort_values(asc
print("\nAverage Speed by Road Type:")
print(speed_by_road)

#Busiest road type (highest volume)

busiest_road = vehicle_by_road.idxmax()
print(f"\nRoad type with highest traffic volume: {busiest_road}")

fastest_road = speed_by_road.idxmax()
print(f"Road type with highest average speed: {fastest_road}")
```

```
Average Vehicle Count by Road Type:
Road_Type
Main_Road         1566.289808
Highway           1283.194444
Secondary_Road     998.472222
Main_Road          380.000000
Name: Vehicle_Count, dtype: float64

Average Speed by Road Type:
Road_Type
Main_Road         48.700000
Secondary_Road    28.252778
Highway           22.292102
Main_Road         17.372727
Name: Average_Speed_kmh, dtype: float64

Road type with highest traffic volume: Main_Road
Road type with highest average speed: Main_Road
```

-📝 Step 5 Summary: Road Type Performance 🚗 Traffic Volume by Road Type:

- The analysis shows that:
- Main_Road has the highest average vehicle count (1,566 vehicles/hour), indicating it carries the majority of daily traffic in Jakarta.
- Highway follows with ~1,283 vehicles/hour.
- Secondary_Road sees the lowest volume at ~998 vehicles/hour.
- ⚠️ Note: There appears to be a duplicate Main_Road entry with a significantly lower volume (380), suggesting potential data entry inconsistencies that may require cleaning.
- 🚀 Average Speed by Road Type:
- Surprisingly:
- Main_Road also shows the highest average speed (48.7 km/h), which is unusual given it also carries the highest volume.
- Secondary_Road follows with 28.25 km/h.
- Highway is the slowest at 22.29 km/h — possibly due to bottlenecks or limited access.
- ⚠️ Again, the second Main_Road entry shows an unusually low speed (17.37 km/h), further confirming potential label duplication.
- 📌 Key Insights:
- Main_Road appears as both the busiest and the fastest road type, though this may be skewed by inconsistent labeling.

- Highway is expected to be faster but shows lower average speed, indicating possible congestion or underperformance.
- ✅ Conclusion:
- "Main_Roads dominate Jakarta's traffic in both volume and speed, but data irregularities suggest a need for label standardization before final conclusions. Highways show lower speeds despite
- being built for efficiency, which may signal congestion issues or infrastructure constraints. Addressing inconsistencies and optimizing traffic flow on highways could improve overall mobility."

In [62]:

```
print(df['Road_Type'].unique())
```

```
['Main_Road' 'Highway' 'Secondary_Road' 'Main_Road ']
```

In [63]:

```
# Strip whitespace and standardize case in 'Road_Type'
df['Road_Type'] = df['Road_Type'].str.strip().str.title().str.replace(' ', '_')

print(df['Road_Type'].unique())
```

```
['Main_Road' 'Highway' 'Secondary_Road']
```

In [64]:

```
#Average number of vehicles per road type
vehicle_by_road = df.groupby('Road_Type')['Vehicle_Count'].mean().sort_values(ascen
print("Average Vehicle Count by Road Type:")
print(vehicle_by_road)

#Average speed per road type
speed_by_road = df.groupby('Road_Type')['Average_Speed_kmh'].mean().sort_values(asc
print("\nAverage Speed by Road Type:")
print(speed_by_road)

#Busiest road type (highest volume)

busiest_road = vehicle_by_road.idxmax()
print(f"\nRoad type with highest traffic volume: {busiest_road}")

fastest_road = speed_by_road.idxmax()
print(f"Road type with highest average speed: {fastest_road}")
```

```
Average Vehicle Count by Road Type:
Road_Type
Main_Road         1558.051684
Highway           1283.194444
Secondary_Road     998.472222
Name: Vehicle_Count, dtype: float64

Average Speed by Road Type:
Road_Type
Secondary_Road    28.252778
Highway           22.292102
Main_Road         17.590278
Name: Average_Speed_kmh, dtype: float64

Road type with highest traffic volume: Main_Road
Road type with highest average speed: Secondary_Road
```

📝 Step 5 Summary: Road Type Performance

🚗 Average Vehicle Count by Road Type:

Main_Road carries the highest traffic volume, with an average of **1,558 vehicles/hour**, highlighting its critical role in daily commuting across Jakarta.

Highway follows with ~**1,283 vehicles/hour**.

Secondary_Road has the lowest traffic volume at ~**998 vehicles/hour**, likely functioning as supporting or alternate routes.

🚀 Average Speed by Road Type:

Secondary_Road records the highest average speed (**28.25 km/h**), indicating smoother flow due to lighter usage or fewer bottlenecks.

Highway surprisingly shows a lower average speed (**22.29 km/h**), possibly due to merging traffic or congestion at access points.

Main_Road has the lowest speed (**17.59 km/h**), despite carrying the highest traffic load — confirming it as the most congested type.

📌 Key Insights:

- **Main_Road** is the busiest but also the **slowest**, suggesting serious congestion that may benefit from targeted traffic control or signal optimization.
- **Secondary_Road** offers the **best performance** in terms of speed, likely due to less volume and fewer intersections.

✅ Conclusion:

> *"Main Roads are essential but heavily congested, requiring traffic relief strategies such as rerouting, adaptive traffic signals, or infrastructure improvements. Secondary Roads perform best in speed and may be promoted as alternative routes to ease Main Road congestion. Meanwhile, the lower-than-expected performance on Highways calls for a deeper review of their efficiency and access design."*

✅ Rush Hour Deep Dive

🎯 Objectives:

** Analyze the "Morning Rush" (7–9) and "Evening Rush" (16–19)

- Find the most congested locations during rush hour
- Compare speed & volume
- Find the most congested days during evening rush

In [65]:

```python
rush_df = df[df['Time_Period'].isin(['Morning Rush', 'Evening Rush'])]


# Most congested location in morning rush
morning_peak = rush_df[rush_df['Time_Period'] == 'Morning Rush']
most_crowded_morning = morning_peak.groupby('Location')['Vehicle_Count'].mean().idx
print(f"🚗 Most congested location during Morning Rush: {most_crowded_morning}")

# Most congested location in evening rush
evening_peak = rush_df[rush_df['Time_Period'] == 'Evening Rush']
most_crowded_evening = evening_peak.groupby('Location')['Vehicle_Count'].mean().idx
print(f"🚗 Most congested location during Evening Rush: {most_crowded_evening}")
```

```
🚗 Most congested location during Morning Rush: Thamrin-Sudirman
🚗 Most congested location during Evening Rush: Thamrin-Sudirman
```

In [66]:

```python
#Average speed in morning vs evening rush
morning_speed = morning_peak['Average_Speed_kmh'].mean()
evening_speed = evening_peak['Average_Speed_kmh'].mean()

print(f"\n🚦 Average Speed during Morning Rush: {morning_speed:.2f} km/h")
print(f"🚦 Average Speed during Evening Rush: {evening_speed:.2f} km/h")

#The most congested day during evening rush
worst_evening_day = evening_peak.groupby('Day_of_Week')['Vehicle_Count'].mean().idx
print(f"📅 Day with the worst Evening Rush traffic: {worst_evening_day}")
```

```
🚦 Average Speed during Morning Rush: 19.79 km/h
🚦 Average Speed during Evening Rush: 16.77 km/h
📅 Day with the worst Evening Rush traffic: Friday

🚦 Average Speed during Evening Rush: 16.77 km/h
📅 Day with the worst Evening Rush traffic: Friday
```

🧠 Final Insights and Recommendations

🔎 Key Insights:

1. **Main_Roads handle the highest traffic volume**, but also record the **lowest average speed**, confirming severe congestion during peak hours.
2. **Evening Rush is more severe** than Morning Rush, with slower speeds and higher vehicle counts — especially on weekdays.
3. **Secondary_Roads offer the best speed performance**, making them potential candidates for traffic redirection or optimization.

🛠️ Recommendations:

1. **Implement adaptive traffic signal systems** on Main_Roads, especially during Evening Rush, to mitigate bottlenecks and improve flow.
2. **Promote the use of Secondary_Roads as alternative routes**, especially for short-distance commuters or non-commercial traffic.

😯 Surprising Insight:

> Despite expectations, **Highways show lower average speed than Secondary Roads**, indicating either congestion at access points or underutilization of express lanes — this requires further investigation into highway design or traffic merging behavior.

🧠 Insight dan Rekomendasi Akhir

🔎 Insight Utama:

1. **Jalan utama (Main_Road)** menampung volume kendaraan tertinggi, namun juga memiliki **kecepatan rata-rata terendah**, menunjukkan tingkat kemacetan yang parah terutama pada jam sibuk.
2. **Jam sibuk sore (Evening Rush)** lebih parah dibandingkan pagi hari, dengan kecepatan yang lebih lambat dan jumlah kendaraan yang lebih banyak — terutama saat hari kerja.
3. **Jalan sekunder (Secondary_Road)** menunjukkan performa terbaik dalam hal kecepatan, sehingga layak dipertimbangkan sebagai jalur alternatif atau jalur

pendukung distribusi lalu lintas.

🛠️ Rekomendasi:

1. **Implementasikan sistem lampu lalu lintas adaptif (adaptive traffic signals)** di jalan utama, khususnya saat jam sibuk sore untuk mengurangi kemacetan dan memperlancar arus lalu lintas.
2. **Promosikan penggunaan jalan sekunder sebagai jalur alternatif**, terutama bagi pengendara jarak pendek atau non-komersial, agar beban jalan utama bisa dikurangi.

😮 Insight Mengejutkan:

> Meskipun secara umum jalan tol (Highway) dianggap lebih cepat, ternyata rata-rata kecepatannya justru lebih rendah dari jalan sekunder — hal ini bisa disebabkan oleh kemacetan di akses masuk/keluar atau desain lalu lintas yang kurang efisien, dan perlu dianalisis lebih lanjut.