In [7]:
```
pip install pandas PyPDF2
```

```
Requirement already satisfied: pandas in /Users/joeynewfield/anaconda3/lib/python3.11/site-packag
es (1.5.3)
Requirement already satisfied: PyPDF2 in /Users/joeynewfield/anaconda3/lib/python3.11/site-packag
es (3.0.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /Users/joeynewfield/anaconda3/lib/python
3.11/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /Users/joeynewfield/anaconda3/lib/python3.11/site-
packages (from pandas) (2022.7)
Requirement already satisfied: numpy>=1.21.0 in /Users/joeynewfield/anaconda3/lib/python3.11/sit
e-packages (from pandas) (1.24.3)
Requirement already satisfied: six>=1.5 in /Users/joeynewfield/anaconda3/lib/python3.11/site-pack
ages (from python-dateutil>=2.8.1->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [6]:
```
import pandas as pd
import numpy as np
from datetime import datetime, timedelta
from PyPDF2 import PdfReader
import re

def read_pdf_content(pdf_path):
    """Read content from PDF file."""
    try:
        reader = PdfReader(pdf_path)
        text_content = ""
        for page in reader.pages:
            text_content += page.extract_text() + "\n"
        return text_content
    except Exception as e:
        print(f"Error reading PDF: {str(e)}")
        return None

def parse_traffic_pdf(text_content):
    """Parse traffic data from the Destin Traffic Data format."""
    if not text_content:
        print("No text content to parse")
        return None

    traffic_data = []
    current_month = None
    current_year = "2023"
    direction = None

    # Split into lines and remove empty lines
    lines = [line.strip() for line in text_content.split('\n') if line.strip()]

    print(f"Total lines in PDF: {len(lines)}")

    for i, line in enumerate(lines):
        # Extract month
        if 'HOURLY CONTINUOUS COUNTS FINAL REPORT' in line:
            for next_line in lines[i:i+3]:
                month_match = re.search(r'(JANUARY|FEBRUARY|MARCH|APRIL|MAY|JUNE|JULY|AUGUST|SEPTE
                if month_match:
                    current_month = month_match.group(1)
```

```python
                    print(f"Found month: {current_month}")
                    break

        # Extract direction
        if 'DIRECTION:' in line:
            direction = line.split('DIRECTION:')[1].split()[0]
            print(f"Found direction: {direction}")

        # Process data rows
        if re.match(r'^\d{1,2}\s+[SMTWRFA]', line):
            parts = line.split()
            try:
                day = int(parts[0])

                # Extract the 24 hourly counts
                hourly_data = parts[2:26]  # Skip day number and day code
                if len(hourly_data) == 24:
                    # Process each hour
                    for hour, count_str in enumerate(hourly_data):
                        # Clean the count string (remove any non-digit characters)
                        count = int(''.join(filter(str.isdigit, count_str)))

                        # Create date string
                        if current_month and direction:
                            month_num = datetime.strptime(current_month, '%B').month
                            date_str = f"{current_year}-{month_num:02d}-{day:02d}"

                            traffic_data.append({
                                'date': date_str,
                                'hour': hour,
                                'traffic_count': count,
                                'direction': direction,
                                'day_code': parts[1]
                            })

            except (ValueError, IndexError) as e:
                continue

    if not traffic_data:
        print("No traffic data was parsed")
        return None
    df = pd.DataFrame(traffic_data)

    print(f"\nParsed data summary:")
    print(f"Total records: {len(df)}")
    print(f"Date range: {df['date'].min()} to {df['date'].max()}")
    print(f"Unique directions: {df['direction'].unique()}")

    daily_totals = df.groupby(['date', 'direction'])['traffic_count'].sum().reset_index()
    daily_totals = daily_totals.rename(columns={'traffic_count': 'daily_total'})
    df = pd.merge(df, daily_totals, on=['date', 'direction'])

    return df

def clean_weather_data(weather_df):
    """Clean and process weather data."""
    try:
        # Convert timestamp - handle the UTC format properly
        weather_df['timestamp'] = pd.to_datetime(
            weather_df['dt_iso'].str.replace(' \+0000 UTC', '', regex=True),
            format='%Y-%m-%d %H:%M:%S'
        )

        # Extract relevant columns and rename for clarity
        cleaned_df = weather_df[[
            'timestamp', 'temp', 'humidity', 'wind_speed',
            'visibility', 'rain_1h', 'weather_main'
        ]].copy()

        # Rename columns
        cleaned_df = cleaned_df.rename(columns={
            'temp': 'temperature',
            'rain_1h': 'precipitation'
        })

        # Handle missing values
        cleaned_df['precipitation'] = cleaned_df['precipitation'].fillna(0)
        cleaned_df = cleaned_df.fillna(method='ffill', limit=3)
        cleaned_df = cleaned_df.dropna()

        # Add date and hour columns for merging
        cleaned_df['date'] = cleaned_df['timestamp'].dt.strftime('%Y-%m-%d')
        cleaned_df['hour'] = cleaned_df['timestamp'].dt.hour
        print("\nWeather data summary:")
        print(f"Total records: {len(cleaned_df)}")
        print(f"Date range: {cleaned_df['date'].min()} to {cleaned_df['date'].max()}")

        return cleaned_df

    except Exception as e:
        print(f"Error cleaning weather data: {str(e)}")
        import traceback
        print(traceback.format_exc())
        return None

def merge_datasets(traffic_df, weather_df):
    """Merge traffic and weather datasets."""
    if traffic_df is None or weather_df is None:
        print("Cannot merge datasets - missing data")
        return None

    try:
        # Merge on date and hour
        merged_df = pd.merge(
            traffic_df,
            weather_df,
            on=['date', 'hour'],
            how='inner'
        )

        # Add temporal features
        merged_df['timestamp'] = pd.to_datetime(merged_df['date']) + pd.to_timedelta(merged_df['ho
        merged_df['day_of_week'] = merged_df['timestamp'].dt.day_name()
        merged_df['is_weekend'] = merged_df['timestamp'].dt.dayofweek.isin([5, 6])
        merged_df['month'] = merged_df['timestamp'].dt.month

        # Reorder columns
        column_order = [
            'timestamp', 'date', 'hour', 'day_of_week', 'is_weekend', 'month',
            'direction', 'traffic_count', 'daily_total', 'temperature', 'humidity',
            'wind_speed', 'visibility', 'precipitation', 'weather_main'
        ]

        merged_df = merged_df[column_order]
        print(f"\nMerged data summary:")
        print(f"Total records: {len(merged_df)}")
        print(f"Date range: {merged_df['date'].min()} to {merged_df['date'].max()}")

        return merged_df

    except Exception as e:
        print(f"Error merging datasets: {str(e)}")
        import traceback
        print(traceback.format_exc())
        return None

def main():
    try:
        # Read the PDF file
        print("Reading PDF file...")
        traffic_text = read_pdf_content('Destin Traffic Data.pdf')

        if traffic_text:
            print("\nPDF read successfully. First 500 characters:")
            print(traffic_text[:500])
            print("...")

        # Parse traffic data
        print("\nParsing traffic data...")
        traffic_df = parse_traffic_pdf(traffic_text)

        if traffic_df is not None:
            # Read and clean weather data
            print("\nProcessing weather data...")
            weather_df = pd.read_csv('Destin Weather Data.csv')
            weather_df = clean_weather_data(weather_df)

            if weather_df is not None:
                # Merge datasets
                print("\nMerging datasets...")
                combined_df = merge_datasets(traffic_df, weather_df)

                if combined_df is not None:
                    # Save to CSV
                    output_file = 'merged_traffic_weather_data.csv'
                    combined_df.to_csv(output_file, index=False)
                    print(f"\nData successfully processed and saved to '{output_file}'")

                    # Display summary statistics
                    print("\nFinal Dataset Summary:")
                    print(f"Total records: {len(combined_df)}")
                    print(f"Date range: {combined_df['date'].min()} to {combined_df['date'].max()}'
                    print(f"Unique directions: {combined_df['direction'].unique()}")
                    print("\nAverage traffic by direction:")
                    print(combined_df.groupby('direction')['traffic_count'].mean())

    except Exception as e:
        print(f"\nError in processing: {str(e)}")
        import traceback
        print(traceback.format_exc())

if __name__ == "__main__":
    main()
```

```
Reading PDF file...

PDF read successfully. First 500 characters:
DATE 05/06/24                                      FLORIDA DEPARTMENT OF TRANSPORTATION
                                                                TRAFFIC COUNTS
                                                    HOURLY CONTINUOUS COUNTS FINAL REPORT
                                                                JANUARY 2023

 COUNTY NAME:  OKALOOSA      STATION:  0386    DIRECTION:  E  LANE:  0
 DESCRIPTION: SR 30/US-98, 300'  W OF CR-30C /BCH DR
 LOCATION: COUNTY 57 SECTION 030 SUBSECTI
...

Parsing traffic data...
Total lines in PDF: 1186
Found month: JANUARY
Found direction: E
Found month: JANUARY
Found direction: W
Found month: FEBRUARY
Found direction: E
Found month: FEBRUARY
Found direction: W
Found month: MARCH
Found direction: E
Found month: MARCH
Found direction: W
Found month: APRIL
Found direction: E
Found month: APRIL
Found direction: W
Found month: MAY
Found direction: E
Found month: MAY
Found direction: W
Found month: JUNE
Found direction: E
Found month: JUNE
Found direction: W
Found month: JULY
Found direction: E
Found month: JULY
Found direction: W
Found month: AUGUST
Found direction: E
Found month: AUGUST
Found direction: W
Found month: SEPTEMBER
Found direction: E
Found month: SEPTEMBER
Found direction: W
Found month: OCTOBER
Found direction: E
Found month: OCTOBER
Found direction: W
Found month: NOVEMBER
Found direction: E
Found month: NOVEMBER
Found direction: W
Found month: DECEMBER
Found direction: E
Found month: DECEMBER
Found direction: W

Parsed data summary:
Total records: 15672
Date range: 2023-01-01 to 2023-12-31
Unique directions: ['E' 'W']

Processing weather data...

Weather data summary:
Total records: 9042
Date range: 2023-01-01 to 2023-12-31

Merging datasets...

Merged data summary:
Total records: 16107
Date range: 2023-01-01 to 2023-12-31

Data successfully processed and saved to 'merged_traffic_weather_data.csv'
```

```
Final Dataset Summary:
Total records: 16107
Date range: 2023-01-01 to 2023-12-31
Unique directions: ['E' 'W']

Average traffic by direction:
direction
E    909.919826
W    890.064153
Name: traffic_count, dtype: float64
```

```
In [ ]:
```