In [2]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn.feature_selection import SelectKBest, f_regression
import seaborn as sns
import matplotlib.pyplot as plt
from scipy import stats

def load_and_prepare_data(file_path):
    """Load and prepare the merged dataset."""
    print("Loading data...")
    df = pd.read_csv(file_path)

    # Convert timestamp to datetime
    df['timestamp'] = pd.to_datetime(df['timestamp'])

    return df

def create_time_features(df):
    """Create time-based features."""
    print("Creating time-based features...")

    # Extract time components
    df['hour_sin'] = np.sin(2 * np.pi * df['hour']/24)
    df['hour_cos'] = np.cos(2 * np.pi * df['hour']/24)

    # Create peak hour flags
    df['is_morning_peak'] = df['hour'].between(6, 9)
    df['is_evening_peak'] = df['hour'].between(16, 19)

    # Add day of week encoding
    df['day_of_week_num'] = pd.to_datetime(df['date']).dt.dayofweek

    # Create weekend dummy (already exists but ensuring numeric)
    df['is_weekend'] = df['is_weekend'].astype(int)

    return df

def create_weather_features(df):
    """Create weather-related features."""
```

```python
    print("Creating weather-related features...")

    # Create weather condition categories
    weather_severity = {
        'Clear': 0,
        'Clouds': 1,
        'Mist': 2,
        'Fog': 3,
        'Rain': 4,
        'Snow': 5,
        'Thunderstorm': 6
    }
    df['weather_severity'] = df['weather_main'].map(weather_severity)

    # Create visibility categories
    df['visibility_category'] = pd.cut(df['visibility'],
                                     bins=[0, 1000, 5000, 10000, float('inf')],
                                     labels=['Very Low', 'Low', 'Moderate', 'Good'])

    # Create temperature categories
    df['temp_category'] = pd.cut(df['temperature'],
                                bins=[-float('inf'), 32, 50, 70, 85, float('inf')],
                                labels=['Freezing', 'Cold', 'Moderate', 'Warm', 'Hot'])

    return df

def create_traffic_features(df):
    """Create traffic-related features."""
    print("Creating traffic-related features...")

    # Calculate rolling averages
    df['rolling_avg_3h'] = df.groupby('direction')['traffic_count'].transform(
        lambda x: x.rolling(window=3, min_periods=1).mean())

    # Calculate traffic density (traffic_count relative to daily_total)
    df['traffic_density'] = df['traffic_count'] / df['daily_total']

    # Create congestion levels
    df['congestion_level'] = pd.qcut(df['traffic_count'],
                                   q=4,
                                   labels=['Low', 'Moderate', 'High', 'Severe'])
```

```
    return df

def select_features(df):
    """Perform feature selection and analysis."""
    print("Analyzing feature importance...")

    # Prepare numeric features for correlation analysis
    numeric_features = df.select_dtypes(include=[np.number]).columns
    numeric_df = df[numeric_features].copy()

    # Calculate correlation with traffic_count
    correlations = numeric_df.corr()['traffic_count'].sort_values(ascending=False)

    # Create correlation heatmap
    plt.figure(figsize=(12, 8))
    sns.heatmap(numeric_df[numeric_features].corr(), annot=True, cmap='coolwarm', center=0)
    plt.title('Feature Correlation Heatmap')
    plt.tight_layout()
    plt.savefig('correlation_heatmap.png')
    plt.close()

    return correlations

def main():
    # Load data
    df = load_and_prepare_data('merged_traffic_weather_data.csv')

    # Feature engineering
    df = create_time_features(df)
    df = create_weather_features(df)
    df = create_traffic_features(df)

    # Feature selection
    correlations = select_features(df)

    # Save engineered features
    output_file = 'engineered_traffic_data.csv'
    df.to_csv(output_file, index=False)
    print(f"\nData saved to {output_file}")

    # Print feature correlations with traffic count
    print("\nFeature Correlations with Traffic Count:")
```

```
    print(correlations)

    # Print summary statistics
    print("\nSummary of Key Features:")
    print(df[['traffic_count', 'rolling_avg_3h', 'traffic_density',
              'temperature', 'visibility', 'wind_speed']].describe())

    # Print unique values in categorical features
    print("\nUnique values in categorical features:")
    categorical_features = ['weather_main', 'visibility_category',
                            'temp_category', 'congestion_level']
    for feature in categorical_features:
        print(f"\n{feature}:")
        print(df[feature].value_counts())

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

```
Loading data...
Creating time-based features...
Creating weather-related features...
Creating traffic-related features...
Analyzing feature importance...

Data saved to engineered_traffic_data.csv

Feature Correlations with Traffic Count:
traffic_count       1.000000
traffic_density     0.959020
rolling_avg_3h      0.949396
hour                0.399000
daily_total         0.233870
temperature         0.070084
humidity            0.047557
wind_speed          0.046900
weather_severity    0.040620
precipitation       0.004272
day_of_week_num    -0.016191
visibility         -0.019112
month              -0.031066
is_weekend         -0.052858
hour_sin           -0.341407
hour_cos           -0.822100
Name: traffic_count, dtype: float64

Summary of Key Features:
       traffic_count  rolling_avg_3h  traffic_density   temperature  \
count   16107.000000    16107.000000     16107.000000  16107.000000
mean      900.273670      900.277416         0.041764     70.566155
std       566.625179      546.606876         0.025816     12.375593
min        23.000000       33.333333         0.001169     31.930000
25%       295.000000      328.666667         0.014198     62.830000
50%      1033.000000     1014.333333         0.049053     72.160000
75%      1420.000000     1411.833333         0.063703     79.660000
max      1918.000000     1842.333333         0.102131    100.220000

         visibility    wind_speed
count  16107.000000  16107.000000
mean    9690.925002      8.674384
```

```
std     1432.086361      4.475895
min      201.000000      0.000000
25%    10000.000000      5.750000
50%    10000.000000      8.050000
75%    10000.000000     11.500000
max    10000.000000     31.070000

Unique values in categorical features:

weather_main:
Clear           8862
Clouds          3151
Rain            2329
Mist             913
Haze             236
Fog              210
Drizzle          190
Thunderstorm     175
Smoke             39
Squall             2
Name: weather_main, dtype: int64

visibility_category:
Moderate    15566
Low           377
Very Low      164
Good            0
Name: visibility_category, dtype: int64

temp_category:
Warm        7384
Moderate    5697
Hot         1883
Cold        1141
Freezing       2
Name: temp_category, dtype: int64

congestion_level:
Low         4034
High        4034
Moderate    4021
Severe      4018
```

```
Name: congestion_level, dtype: int64
```

In [ ]:

```
```