In [3]:
```
import pandas as pd
import numpy as np
from scipy import stats
from datetime import datetime, timedelta
from typing import Dict, List, Tuple, Optional
import matplotlib.pyplot as plt
import seaborn as sns
import calendar

# Define MCC (Merchant Category Code) mappings
MCC_CATEGORIES = {
    '5411': 'Grocery Stores',
    '5812': 'Restaurants',
    '5541': 'Gas Stations',
    '5311': 'Department Stores',
    '5999': 'Miscellaneous Retail'
}

def analyze_settlement_data(transactions: List[Dict]) -> Tuple[pd.DataFrame, pd.DataFr
    """
    Analyze settlement data to generate comprehensive spending analysis.
    """
    if not transactions:
        raise ValueError("No transactions provided")

    # Convert transaction list to DataFrame
    df = pd.DataFrame(transactions)

    # Validate required columns
    required_columns = ['transaction_id', 'transaction_date', 'merchant_name', 'mcc',
    missing_columns = [col for col in required_columns if col not in df.columns]
    if missing_columns:
        raise ValueError(f"Missing required columns: {missing_columns}")

    # Clean and standardize the data
    df['amount'] = pd.to_numeric(df['amount'], errors='coerce')
    df['transaction_date'] = pd.to_datetime(df['transaction_date'], errors='coerce')
    df = df.dropna(subset=['amount', 'transaction_date'])

    if df.empty:
        raise ValueError("No valid transactions after cleaning data")

    # Map MCC codes to categories
    df['category'] = df['mcc'].map(MCC_CATEGORIES).fillna('Other')

    # Generate analysis components
    category_summary = generate_category_summary(df)
    daily_trends = generate_daily_trends(df)
    insights = generate_insights(df, category_summary)

    return category_summary, daily_trends, insights

def generate_category_summary(df: pd.DataFrame) -> pd.DataFrame:
    """Generate category-wise spending summary."""
    summary = df.groupby('category').agg({
        'amount': ['sum', 'count', 'mean', 'std'],
        'transaction_date': ['min', 'max']
    }).round(2)
    summary.columns = ['total_amount', 'transaction_count',
                      'average_transaction', 'std_deviation',
                      'first_transaction', 'last_transaction']
    summary = summary.reset_index()

    # Add derived metrics
    total_spend = summary['total_amount'].sum()
    summary['spend_percentage'] = (summary['total_amount'] / total_spend * 100).round(

    date_range = (summary['last_transaction'].max() -
                  summary['first_transaction'].min()).days + 1
    summary['transactions_per_day'] = (summary['transaction_count'] /
                                      max(date_range, 1)).round(3)

    return summary

def generate_daily_trends(df: pd.DataFrame) -> pd.DataFrame:
    """Generate daily spending trends and patterns."""
    daily = df.groupby(['transaction_date', 'category']).agg({
        'amount': ['sum', 'count']
    }).round(2)

    daily.columns = ['daily_total', 'daily_transactions']
    daily = daily.reset_index()
    daily['day_of_week'] = daily['transaction_date'].dt.day_name()

    # Add 7-day moving average
    daily['7day_moving_avg'] = (daily.groupby('category')['daily_total']
                               .transform(lambda x: x.rolling(7, min_periods=1).mean()

    return daily

def generate_insights(df: pd.DataFrame, category_summary: pd.DataFrame) -> Dict:
    """Generate key insights from the transaction data."""
    insights = {
        'summary_metrics': {
            'total_spend': df['amount'].sum(),
            'total_transactions': len(df),
            'average_transaction': df['amount'].mean(),
            'unique_merchants': df['merchant_name'].nunique()
        },
        'top_merchants': df.groupby('merchant_name').agg({
            'amount': 'sum',
            'transaction_id': 'count'
        }).sort_values('amount', ascending=False).head(5).to_dict(),
        'spending_patterns': {
            'highest_spending_day': df.groupby(df['transaction_date'].dt.day_name())['
            'highest_spending_category': category_summary.loc[category_summary['total_
            'most_frequent_category': category_summary.loc[category_summary['transacti
        }
    }
    return insights

def plot_category_distribution(category_summary: pd.DataFrame, save_path: Optional[str
    """Plot category-wise spending distribution."""
    plt.figure(figsize=(10, 6))
    plt.pie(category_summary['total_amount'],
            labels=category_summary['category'],
            autopct='%1.1f%%')
    plt.title('Spending Distribution by Category')

    if save_path:
        plt.savefig(f"{save_path}_category_dist.png", bbox_inches='tight')
    plt.close()

def plot_daily_trends(daily_trends: pd.DataFrame, save_path: Optional[str] = None) ->
    """Plot daily spending trends."""
    plt.figure(figsize=(12, 6))

    for category in daily_trends['category'].unique():
        data = daily_trends[daily_trends['category'] == category]
        plt.plot(data['transaction_date'], data['7day_moving_avg'],
                label=f'{category} (7-day avg)')

    plt.title('Daily Spending Trends by Category')
    plt.xlabel('Date')
    plt.ylabel('Amount ($)')
    plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
    plt.tight_layout()

    if save_path:
        plt.savefig(f"{save_path}_daily_trends.png", bbox_inches='tight')
    plt.close()

def plot_weekday_patterns(daily_trends: pd.DataFrame, save_path: Optional[str] = None)
    """Plot spending patterns by day of week."""
    plt.figure(figsize=(10, 6))

    # Ensure days are in correct order
    day_order = list(calendar.day_name)
    daily_trends['day_of_week'] = pd.Categorical(
        daily_trends['day_of_week'],
        categories=day_order,
        ordered=True
    )

    sns.boxplot(data=daily_trends, x='day_of_week', y='daily_total')
    plt.title('Spending Patterns by Day of Week')
    plt.xlabel('Day of Week')
    plt.ylabel('Daily Total ($)')
    plt.xticks(rotation=45)
    plt.tight_layout()

    if save_path:
        plt.savefig(f"{save_path}_weekday_patterns.png", bbox_inches='tight')
    plt.close()

def generate_report(category_summary: pd.DataFrame, insights: Dict) -> str:
    """Generate a text report of the analysis findings."""
    report = []

    report.append("# Transaction Analysis Report")

    # Overall Summary
    report.append("\n## Overall Summary")
    report.append(f"Total Spend: ${insights['summary_metrics']['total_spend']:,.2f}")
    report.append(f"Total Transactions: {insights['summary_metrics']['total_transactic
    report.append(f"Average Transaction: ${insights['summary_metrics']['average_transa
    report.append(f"Unique Merchants: {insights['summary_metrics']['unique_merchants']
    # Category Analysis
    report.append("\n## Category Analysis")
    for _, row in category_summary.sort_values('total_amount', ascending=False).iterro
        report.append(f"\n### {row['category']}")
        report.append(f"Total Spend: ${row['total_amount']:,.2f} ({row['spend_percenta
        report.append(f"Transaction Count: {row['transaction_count']}")
        report.append(f"Average Transaction: ${row['average_transaction']:,.2f}")

    # Top Merchants
    report.append("\n## Top Merchants")
    for merchant, amount in insights['top_merchants']['amount'].items():
        report.append(f"{merchant}: ${amount:,.2f}")

    return "\n".join(report)

def main(
    transaction_data: List[Dict],
    output_path: Optional[str] = None
) -> Tuple[pd.DataFrame, pd.DataFrame, Dict, str]:
    """Main function to run the analysis pipeline."""
    try:
        # Run analysis
        category_summary, daily_trends, insights = analyze_settlement_data(transactior

        # Generate report
        report_text = generate_report(category_summary, insights)

        # Create visualizations
        if output_path:
            plot_category_distribution(category_summary, output_path)
            plot_daily_trends(daily_trends, output_path)
            plot_weekday_patterns(daily_trends, output_path)

            # Save report
            with open(f"{output_path}_report.md", 'w') as f:
                f.write(report_text)

        return category_summary, daily_trends, insights, report_text

    except Exception as e:
        print(f"Error in analysis pipeline: {str(e)}")
        raise

if __name__ == "__main__":
    # Create meaningful sample data
    sample_transactions = [
        {
            "transaction_id": f"t{i}",
            "transaction_date": (datetime(2024, 1, 1) + timedelta(days=i)).strftime("%
            "merchant_name": merchant,
            "mcc": mcc,
            "amount": amount,
            "reference_number": f"ref{i}"
        }
        for i, (merchant, mcc, amount) in enumerate([
            ("Walmart", "5411", 50.25),
            ("Target", "5411", 75.50),
            ("Shell Gas", "5541", 45.00),
            ("Chevron", "5541", 40.00),
            ("Applebees", "5812", 65.30),
            ("McDonalds", "5812", 12.50),
            ("Macys", "5311", 120.75),
            ("Best Buy", "5999", 199.99),
            ("Walmart", "5411", 82.34),
            ("Shell Gas", "5541", 48.50),
            ("Target", "5411", 92.45),
            ("Applebees", "5812", 45.80),
            ("McDonalds", "5812", 15.25),
            ("Best Buy", "5999", 299.99),
            ("Walmart", "5411", 67.82),
            ("Chevron", "5541", 42.50),
            ("Macys", "5311", 89.99),
            ("Target", "5411", 105.67),
            ("Applebees", "5812", 78.90),
            ("Shell Gas", "5541", 44.25),
            # Add more transactions with different dates
        ])
    ]

    # Create output directory if it doesn't exist
    import os
    output_dir = "./analysis_output"
    os.makedirs(output_dir, exist_ok=True)

    # Run analysis
    try:
        category_summary, daily_trends, insights, report = main(
            sample_transactions,
            output_path=os.path.join(output_dir, "analysis")
        )

        print("Analysis completed successfully!")
        print("\nGenerated files:")
        print(f"- {output_dir}/analysis_category_dist.png")
        print(f"- {output_dir}/analysis_daily_trends.png")
        print(f"- {output_dir}/analysis_weekday_patterns.png")
        print(f"- {output_dir}/analysis_report.md")

        print("\nSummary of findings:")
        print(f"Total transactions: {insights['summary_metrics']['total_transactions']
        print(f"Total spend: ${insights['summary_metrics']['total_spend']:,.2f}")
        print(f"Average transaction: ${insights['summary_metrics']['average_transactic

    except Exception as e:
        print(f"Error running analysis: {str(e)}")
```

```
Analysis completed successfully!

Generated files:
- ./analysis_output/analysis_category_dist.png
- ./analysis_output/analysis_daily_trends.png
- ./analysis_output/analysis_weekday_patterns.png
- ./analysis_output/analysis_report.md

Summary of findings:
Total transactions: 20
Total spend: $1,622.75
Average transaction: $81.14
```

In [ ]:

```
```