In [1]:

```
import pandas as pd
import numpy as np
from datetime import datetime, timedelta

# لضبط العشوائية للتكرار
np.random.seed(42)

# إعدادات حجم البيانات
num_customers = 300
num_products = 50
num_orders = 1000

# بيانات العملاء
cities = ['الرياض', 'جدة', 'الدمام', 'المدينة', 'أبها']
genders = ['ذكر', 'أنثى']

customers = pd.DataFrame({
    'customer_id': range(1, num_customers + 1),
    'gender': np.random.choice(genders, size=num_customers),
    'city': np.random.choice(cities, size=num_customers),
    'join_date': [datetime(2022, 1, 1) + timedelta(days=int(np.random.rand()*730)) fo
})

# بيانات المنتجات
categories = ['ملابس', 'إلكترونيات', 'أدوات منزلية', 'مكياج', 'كتب']

products = pd.DataFrame({
    'product_id': range(1, num_products + 1),
    'category': np.random.choice(categories, size=num_products),
    'price': np.random.randint(50, 1000, size=num_products),
    'on_sale': np.random.choice([True, False], size=num_products),
    'rating': np.round(np.random.uniform(2.5, 5.0, size=num_products), 1)
})

# بيانات الطلبات
orders = pd.DataFrame({
    'order_id': range(1, num_orders + 1),
    'order_date': [datetime(2023, 1, 1) + timedelta(days=int(np.random.rand()*580)) +
    'customer_id': np.random.choice(customers['customer_id'], size=num_orders),
    'product_id': np.random.choice(products['product_id'], size=num_orders),
    'quantity': np.random.randint(1, 4, size=num_orders)
})

# إضافة سعر المنتج لكل طلب وحساب السعر الكلي
orders = orders.merge(products[['product_id', 'price']], on='product_id')
orders['total_price'] = orders['price'] * orders['quantity']

print("تم توليد البيانات بنجاح!")
```

```
تم توليد البيانات بنجاح!
```

In [2]:

```
import matplotlib.pyplot as plt
import seaborn as sns

sns.set(style="whitegrid")
plt.rcParams['font.family'] = 'Arial'  # لضمان عرض الخطوط العربية بشكل جيد

# عرض أول 5 صفوف من الجداول
print("Customers Table:")
display(customers.head())

print("Products Table:")
display(products.head())

print("Orders Table:")
display(orders.head())

# استخراج الشهر من تاريخ الطلب
orders['order_month'] = orders['order_date'].dt.to_period('M').astype(str)

# المبيعات الشهرية
monthly_sales = orders.groupby('order_month')['total_price'].sum().reset_index()

plt.figure(figsize=(12, 6))
sns.lineplot(data=monthly_sales, x='order_month', y='total_price', marker='o', color=
plt.title('Monthly Sales')
plt.xlabel('Month')
plt.ylabel('Total Sales (SAR)')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()

# دمج الطلبات مع العملاء لأخذ المدينة
orders_with_city = orders.merge(customers[['customer_id', 'city']], on='customer_id')

city_sales = orders_with_city.groupby('city')['total_price'].sum().sort_values(ascend

plt.figure(figsize=(8, 5))
sns.barplot(x=city_sales.values, y=city_sales.index, palette='Blues_d')
plt.title('Sales by City')
plt.xlabel('Total Sales')
plt.ylabel('City')
plt.show()

# دمج الطلبات مع المنتجات لأخذ الفئة
orders_with_category = orders.merge(products[['product_id', 'category']], on='product

category_sales = orders_with_category.groupby('category')['total_price'].sum().sort_

plt.figure(figsize=(8, 5))
sns.barplot(x=category_sales.values, y=category_sales.index, palette='viridis')
plt.title('Sales by Product Category')
plt.xlabel('Total Sales')
plt.ylabel('Category')
plt.show()

# أفضل 10 عملاء حسب الإنفاق
top_customers = orders.groupby('customer_id')['total_price'].sum().sort_values(ascend
```

```
plt.figure(figsize=(8, 5))
sns.barplot(x=top_customers.values, y=top_customers.index.astype(str), palette='magma
plt.title('Top 10 Customers by Spending')
plt.xlabel('Total Spending')
plt.ylabel('Customer ID')
plt.show()
```

Customers Table:

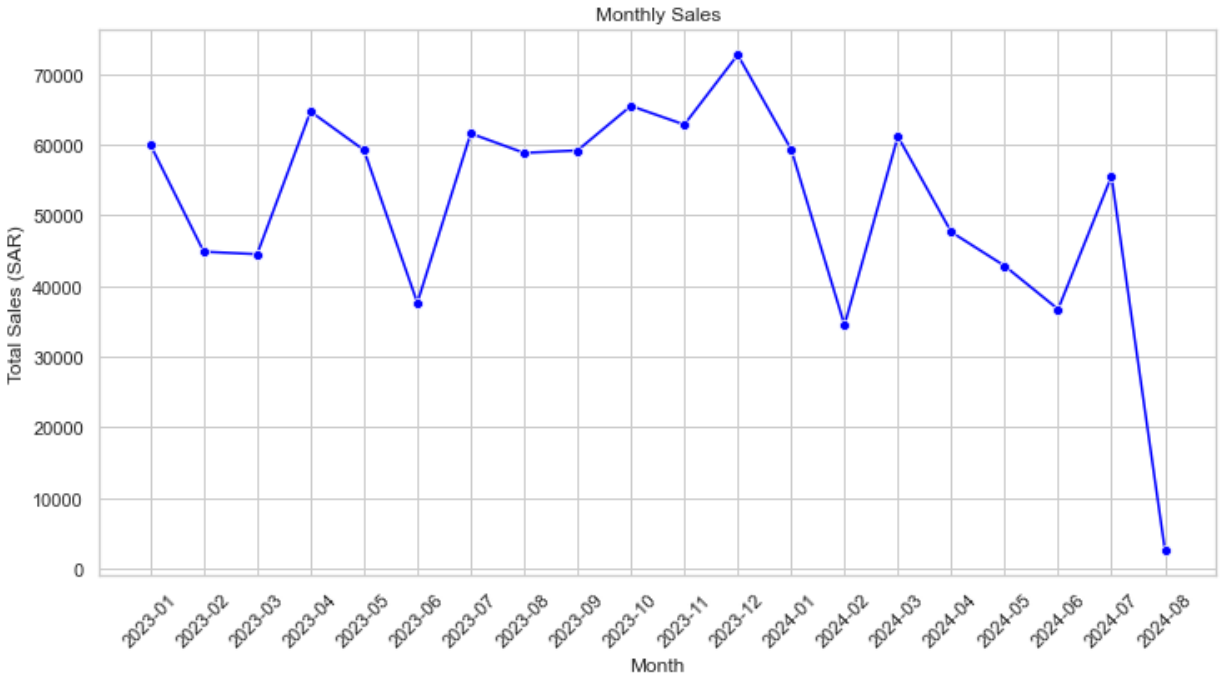
	customer_id	gender	city	join_date
0	1	ذكر	الرياض	2023-06-19
1	2	أنثى	الرياض	2022-04-04
2	3	ذكر	الدمام	2022-07-02
3	4	ذكر	جدة	2023-02-28
4	5	ذكر	أبها	2023-09-25

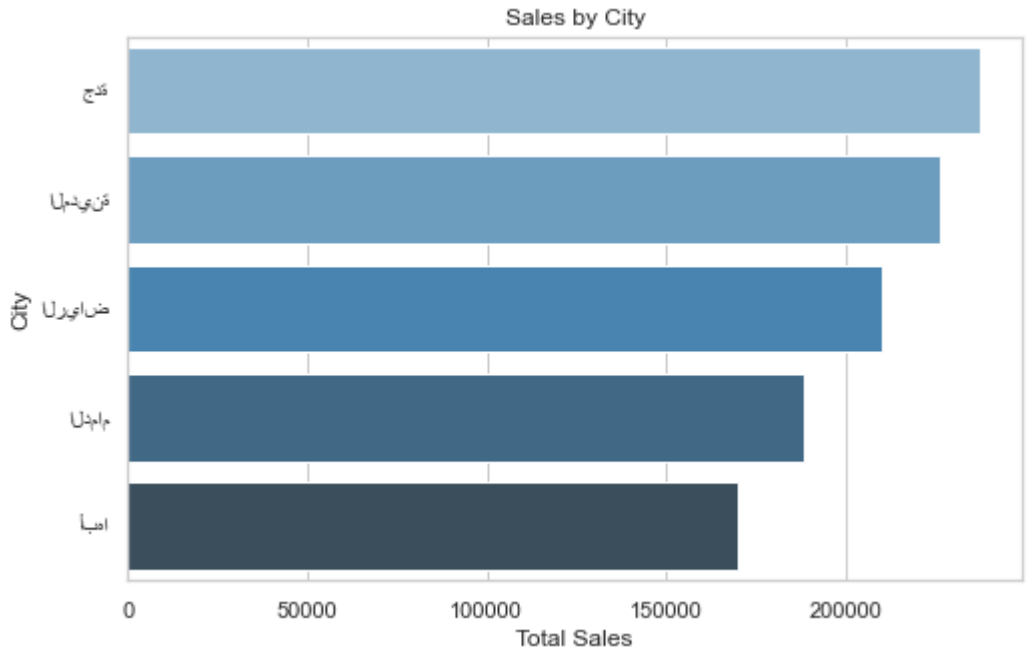
Products Table:

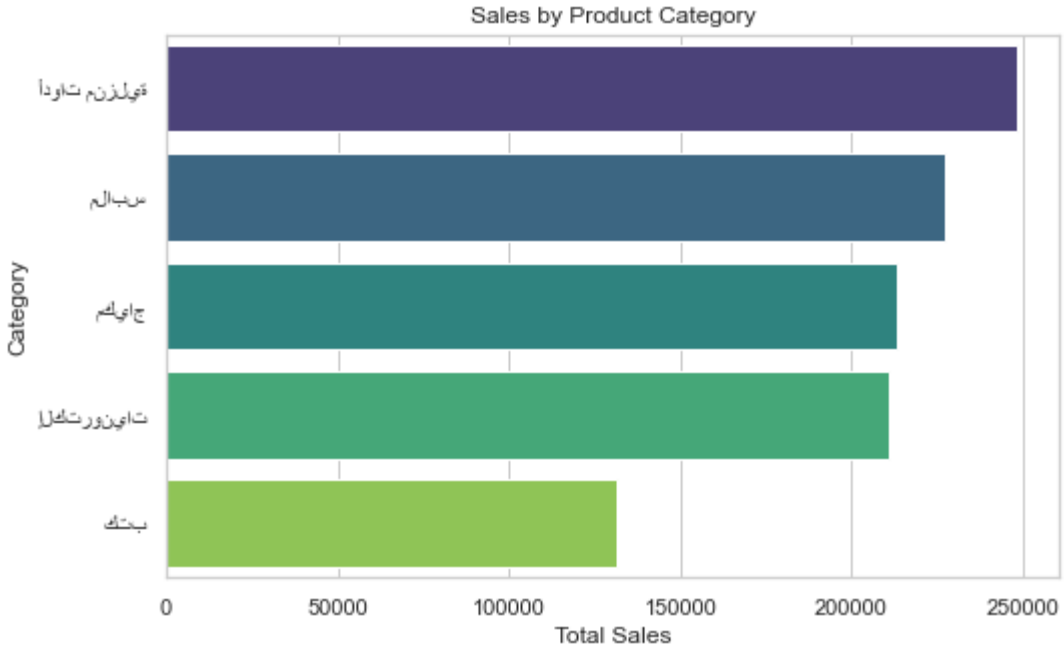
	product_id	category	price	on_sale	rating
0	1	مكياج	319	True	3.9
1	2	أدوات منزلية	877	True	4.3
2	3	مكياج	847	True	4.0
3	4	ملابس	340	False	4.6
4	5	مكياج	774	False	4.9

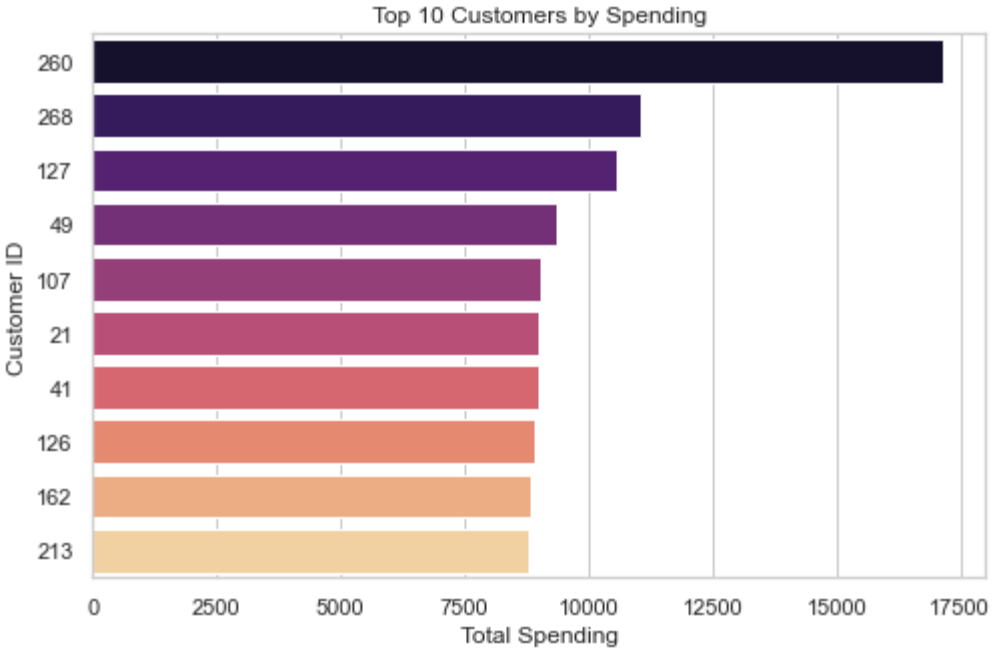
Orders Table:

	order_id	order_date	customer_id	product_id	quantity	price	total_price
0	1	2023-01-20	258	28	2	914	1828
1	6	2023-01-20	288	28	1	914	914
2	49	2023-11-16	283	28	2	914	1828
3	66	2023-08-24	281	28	2	914	1828
4	86	2024-05-14	234	28	3	914	2742


Monthly Sales
Total Sales (SAR)
70000
60000
50000
40000
30000
20000
10000
0
2023-01
2023-02
2023-03
2023-04
2023-05
2023-06
2023-07
2023-08
2023-09
2023-10
2023-11
2023-12
2024-01
2024-02
2024-03
2024-04
2024-05
2024-06
2024-07
2024-08
Month


Sales by City
City
جدة
المدينة
الرياض
الدمام
ابها
0
50000
100000
150000
200000
Total Sales


Sales by Product Category
أدوات منزلية
ملابس
مكياج
إلكترونيات
كتب
Category
0
50000
100000
150000
200000
250000
Total Sales


Top 10 Customers by Spending
260
268
127
49
107
21
41
126
162
213
Customer ID
0
2500
5000
7500
10000
12500
15000
17500
Total Spending

In [3]:

```
import datetime as dt

# نفترض اليوم الأخير في البيانات) نحدد تاريخ اليوم الذي سنحسب منه recency
reference_date = orders['order_date'].max() + pd.Timedelta(days=1)

# إنشاء جدول RFM لكل عميل
rfm = orders.groupby('customer_id').agg({
    'order_date': lambda x: (reference_date - x.max()).days,  # Recency: الفرق بالأيام
    'order_id': 'count',                                      # Frequency: عدد الطلبات
    'total_price': 'sum'                                      # Monetary: مجموع الإنفاق
}).reset_index()

# إعادة تسمية الأعمدة
rfm.columns = ['customer_id', 'Recency', 'Frequency', 'Monetary']

# عرض أول 5 صفوف
display(rfm.head())
```

	customer_id	Recency	Frequency	Monetary
0	1	68	3	3066
1	2	447	3	2544
2	3	52	4	2164
3	4	156	2	632
4	5	87	3	689

In [4]:

```
# تصنيف Recency: (أفضل 5) كل ما كان أقل (أحدث) يحصل على درجة أعلى
rfm['R_Score'] = pd.qcut(rfm['Recency'], 5, labels=range(5, 0, -1))

# تصنيف Frequency: (أفضل 5) كل ما كان أعلى يحصل على درجة أعلى
rfm['F_Score'] = pd.qcut(rfm['Frequency'].rank(method='first'), 5, labels=range(1, 6)

# تصنيف Monetary: (أفضل 5) كل ما كان أعلى يحصل على درجة أعلى
rfm['M_Score'] = pd.qcut(rfm['Monetary'], 5, labels=range(1, 6))

# دمج الدرجات في عمود واحد لتسهيل التصنيف
rfm['RFM_Score'] = rfm['R_Score'].astype(str) + rfm['F_Score'].astype(str) + rfm['M_S

# عرض أول 10 عملاء مع تصنيف RFM
display(rfm.head(10))
```

	customer_id	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	RFM_Score
0	1	68	3	3066	4	2	3	423
1	2	447	3	2544	1	2	2	122
2	3	52	4	2164	4	3	2	432
3	4	156	2	632	2	1	1	211
4	5	87	3	689	3	2	1	321

	customer_id	Recency	Frequency	Monetary	R_Score	F_Score	M_Score	RFM_Score
5	7	40	2	2404	4	1	2	412
6	8	79	5	4974	4	4	4	444
7	10	102	3	1229	3	2	1	321
8	11	128	2	1393	3	1	1	311
9	12	128	4	3394	3	3	3	333

In [5]:

```
def rfm_segment(row):
    if row['R_Score'] >= 4 and row['F_Score'] >= 4 and row['M_Score'] >= 4:
        return 'Champions'
    elif row['R_Score'] >= 3 and row['F_Score'] >= 3 and row['M_Score'] >= 3:
        return 'Loyal Customers'
    elif row['R_Score'] >= 4 and row['F_Score'] <= 2:
        return 'Potential Loyalists'
    elif row['R_Score'] <= 2 and row['F_Score'] >= 4:
        return 'At Risk'
    elif row['R_Score'] <= 2 and row['F_Score'] <= 2:
        return 'Lost'
    else:
        return 'Others'

rfm['Segment'] = rfm.apply(rfm_segment, axis=1)

# عرض عدد العملاء في كل فئة
segment_counts = rfm['Segment'].value_counts()
print(segment_counts)

# رسم بياني لتوزيع الفئات
plt.figure(figsize=(8,5))
sns.barplot(x=segment_counts.index, y=segment_counts.values, palette='Set2')
plt.title('Customer Segments Distribution')
plt.xlabel('Segment')
plt.ylabel('Number of Customers')
plt.show()
```

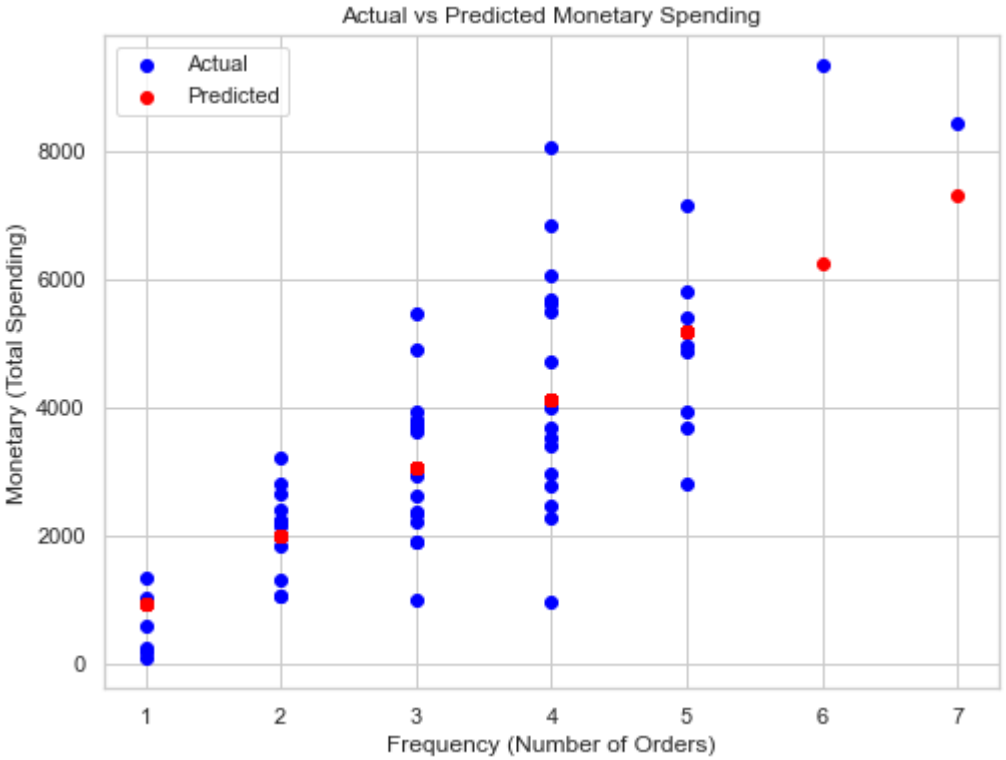
```
Lost                 66
Loyal Customers      60
Others               53
Champions            48
Potential Loyalists  34
At Risk              28
Name: Segment, dtype: int64
```

In [6]:

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score

# إعداد البيانات
X = rfm[['Frequency']]  # المتغير المستقل (عدد الطلبات)
y = rfm['Monetary']     # المتغير التابع (الإنفاق الإجمالي)

# تقسيم البيانات (80% تدريب - 20% اختبار)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state

# بناء نموذج الانحدار الخطي
model = LinearRegression()
model.fit(X_train, y_train)

# التنبؤ على بيانات الاختبار
y_pred = model.predict(X_test)

# تقييم النموذج
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse:.2f}')
print(f'R-squared: {r2:.2f}')

# رسم النتائج
plt.figure(figsize=(8,6))
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.scatter(X_test, y_pred, color='red', label='Predicted')
plt.title('Actual vs Predicted Monetary Spending')
plt.xlabel('Frequency (Number of Orders)')
plt.ylabel('Monetary (Total Spending)')
plt.legend()
plt.show()
```

```
Mean Squared Error: 1838115.86
```

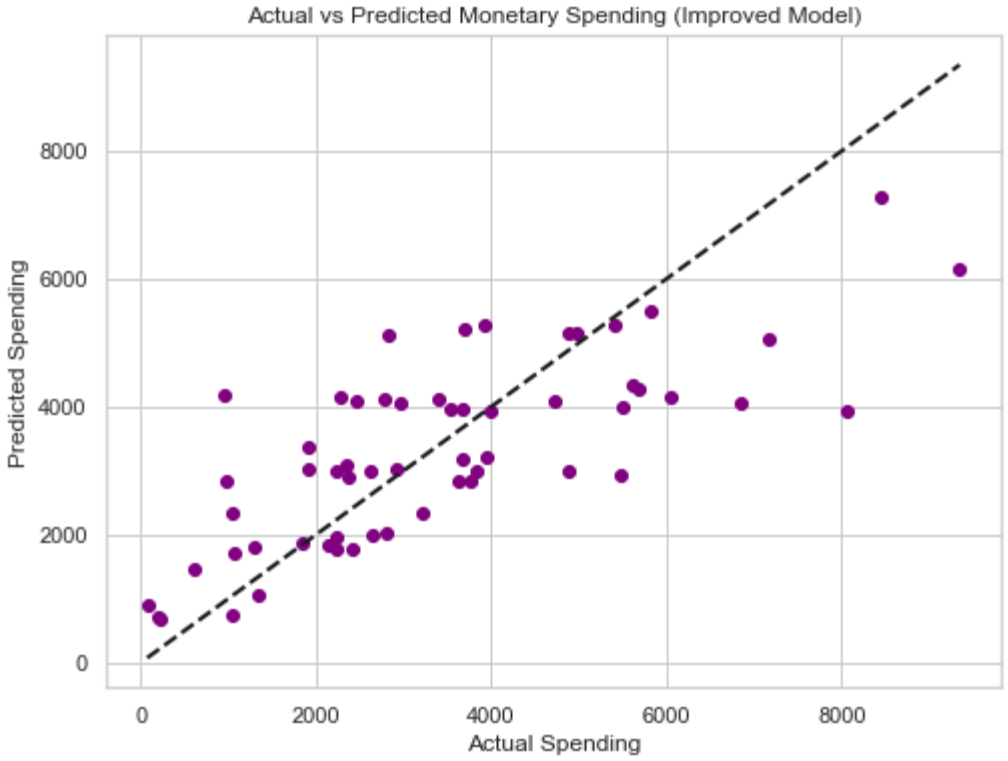
```
R-squared: 0.58
```



In [7]:

```
# المتغيرات المستقلة الجديدة
X = rfm[['Frequency', 'Recency']]
y = rfm['Monetary']

# تقسيم البيانات 80% تدريب و20% اختبار
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state

# بناء النموذج
model = LinearRegression()
model.fit(X_train, y_train)

# التنبؤ على بيانات الاختبار
y_pred = model.predict(X_test)

# تقييم النموذج
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Improved Model Mean Squared Error: {mse:.2f}')
print(f'Improved Model R-squared: {r2:.2f}')

# رسم التوقعات مقابل القيم الحقيقية
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, color='purple')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.title('Actual vs Predicted Monetary Spending (Improved Model)')
plt.xlabel('Actual Spending')
plt.ylabel('Predicted Spending')
plt.show()
```

```
Improved Model Mean Squared Error: 1882391.42
Improved Model R-squared: 0.57
```

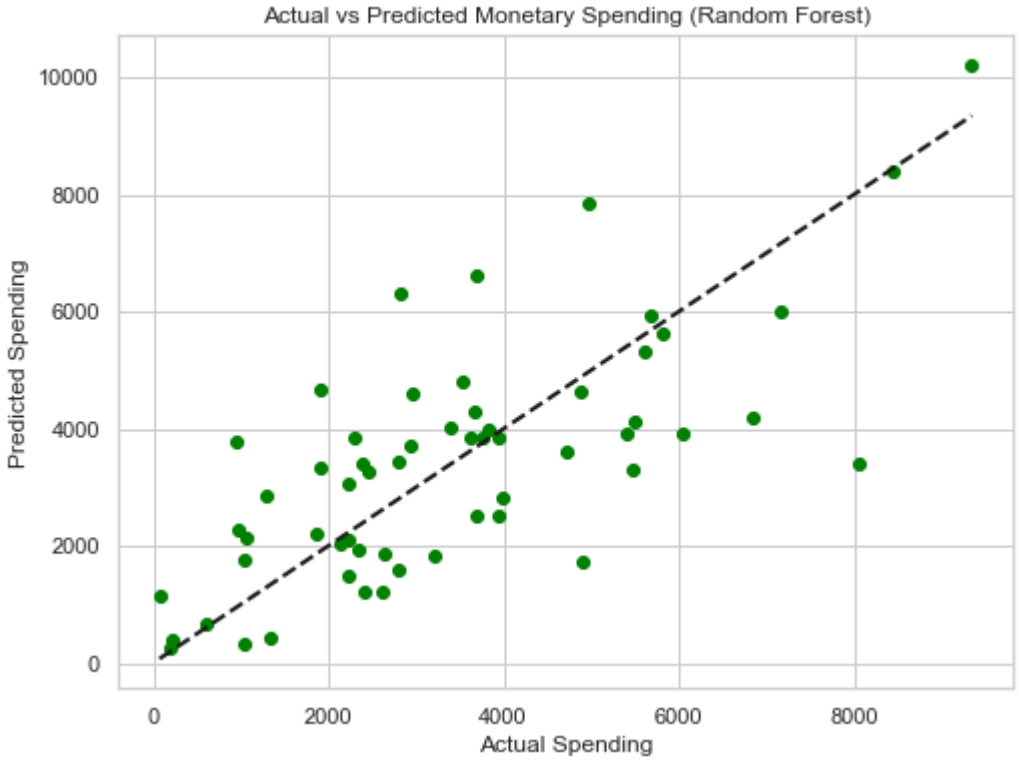


In [8]:

```
from sklearn.ensemble import RandomForestRegressor

# إعداد المتغيرات المستقلة والمتغير التابع
X = rfm[['Frequency', 'Recency']]
y = rfm['Monetary']

# تقسيم البيانات (80% تدريب، 20% اختبار)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state

# بناء نموذج الغابات العشوائية
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# التنبؤ على بيانات الاختبار
y_pred_rf = rf_model.predict(X_test)

# تقييم النموذج
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

print(f'Random Forest MSE: {mse_rf:.2f}')
print(f'Random Forest R-squared: {r2_rf:.2f}')

# رسم القيم الفعلية مقابل المتوقعة
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred_rf, color='green')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.title('Actual vs Predicted Monetary Spending (Random Forest)')
plt.xlabel('Actual Spending')
plt.ylabel('Predicted Spending')
plt.show()
```

```
Random Forest MSE: 2329977.06
```

```
Random Forest R-squared: 0.46
```

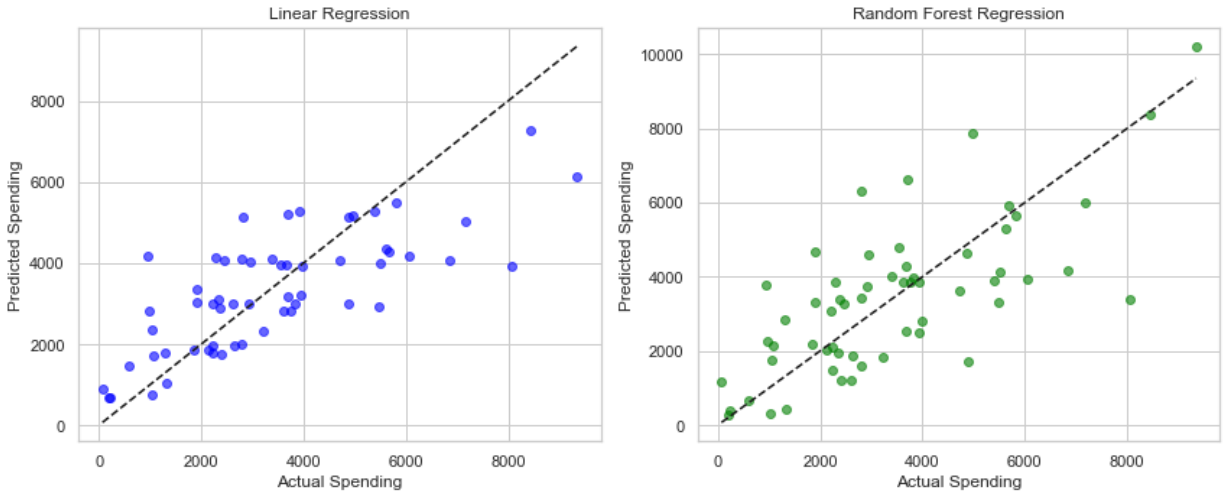


In [9]:

```
# استيراد المكتبات
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt

# تحضير البيانات
X = rfm[['Frequency', 'Recency']]
y = rfm['Monetary']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state

# نموذج الانحدار الخطي
lr_model = LinearRegression()
lr_model.fit(X_train, y_train)
y_pred_lr = lr_model.predict(X_test)
mse_lr = mean_squared_error(y_test, y_pred_lr)
r2_lr = r2_score(y_test, y_pred_lr)

# نموذج الغابات العشوائية
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
y_pred_rf = rf_model.predict(X_test)
mse_rf = mean_squared_error(y_test, y_pred_rf)
r2_rf = r2_score(y_test, y_pred_rf)

# طباعة النتائج
print("Linear Regression:")
print(f"  Mean Squared Error: {mse_lr:.2f}")
print(f"  R-squared: {r2_lr:.2f}\n")

print("Random Forest Regression:")
print(f"  Mean Squared Error: {mse_rf:.2f}")
print(f"  R-squared: {r2_rf:.2f}")

# رسم النتائج
plt.figure(figsize=(12,5))

plt.subplot(1, 2, 1)
plt.scatter(y_test, y_pred_lr, color='blue', alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--')
plt.title('Linear Regression')
plt.xlabel('Actual Spending')
plt.ylabel('Predicted Spending')

plt.subplot(1, 2, 2)
plt.scatter(y_test, y_pred_rf, color='green', alpha=0.6)
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--')
plt.title('Random Forest Regression')
plt.xlabel('Actual Spending')
plt.ylabel('Predicted Spending')

plt.tight_layout()
plt.show()
```

```
Linear Regression:
```

```
  Mean Squared Error: 1882391.42
  R-squared: 0.57

Random Forest Regression:
  Mean Squared Error: 2329977.06
  R-squared: 0.46
```



In [ ]: