In [5]:

```
import pandas as pd
import numpy as np
supply_data=pd.read_csv('supply_chain_data.csv')
supply_data.head()
```

Out[5]:

	Product type	SKU	Price	Availability	Number of products sold	Revenue generated	Customer demographics	Stock levels	Lead times	Order quantities
0	haircare	SKU0	69.808006	55	802	8661.996792	Non-binary	58	7	96
1	skincare	SKU1	14.843523	95	736	7460.900065	Female	53	30	37
2	haircare	SKU2	11.319683	34	8	9577.749626	Unknown	1	10	88
3	skincare	SKU3	61.163343	68	83	7766.836426	Non-binary	23	13	59
4	skincare	SKU4	4.805496	26	871	2686.505152	Non-binary	5	3	56

5 rows × 24 columns

In [7]:

```
missing=supply_data.isnull().sum()
missing
```

Out[7]:

```
Product type               0
SKU                        0
Price                      0
Availability               0
Number of products sold    0
Revenue generated          0
Customer demographics      0
Stock levels               0
Lead times                 0
Order quantities           0
Shipping times             0
Shipping carriers          0
Shipping costs             0
Supplier name              0
Location                   0
Lead time                  0
Production volumes         0
Manufacturing lead time    0
Manufacturing costs        0
Inspection results         0
Defect rates               0
Transportation modes       0
Routes                     0
Costs                      0
dtype: int64
```

In [9]:

```
supply_data.dtypes
```

Out[9]:
```
Product type                  object
SKU                           object
Price                        float64
Availability                   int64
Number of products sold        int64
Revenue generated            float64
Customer demographics         object
Stock levels                   int64
Lead times                     int64
Order quantities               int64
Shipping times                 int64
Shipping carriers             object
Shipping costs               float64
Supplier name                 object
Location                      object
Lead time                      int64
Production volumes             int64
Manufacturing lead time        int64
Manufacturing costs          float64
Inspection results            object
Defect rates                 float64
Transportation modes          object
Routes                        object
Costs                        float64
dtype: object
```

In [11]:
```
duplicate=supply_data.duplicated()
supply_data[duplicate]
```

Out[11]:

Product type	SKU	Price	Availability	Number of products sold	Revenue generated	Customer demographics	Stock levels	Lead times	Order quantities	...	Lo

0 rows × 24 columns

In [16]:
```
round(supply_data.describe(),2)
```

Out[16]:

	Price	Availability	Number of products sold	Revenue generated	Stock levels	Lead times	Order quantities	Shipping times	Shipping costs	Lead time
count	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00	100.00
mean	49.46	48.40	460.99	5776.05	47.77	15.96	49.22	5.75	5.55	17.08
std	31.17	30.74	303.78	2732.84	31.37	8.79	26.78	2.72	2.65	8.85
min	1.70	1.00	8.00	1061.62	0.00	1.00	1.00	1.00	1.01	1.00
25%	19.60	22.75	184.25	2812.85	16.75	8.00	26.00	3.75	3.54	10.00
50%	51.24	43.50	392.50	6006.35	47.50	17.00	52.00	6.00	5.32	18.00
75%	77.20	75.00	704.25	8253.98	73.00	24.00	71.25	8.00	7.60	25.00
max	99.17	100.00	996.00	9866.47	100.00	30.00	96.00	10.00	9.93	30.00

In [13]:
```python
# Check for duplicated column names
duplicated_columns = supply_data.columns[supply_data.columns.duplicated()].tolist()

# Check for missing values and column data types
missing_values = supply_data.isnull().sum()
data_types = supply_data.dtypes

duplicated_columns, missing_values[missing_values > 0], data_types
```

Out[13]:
```
([],
 Series([], dtype: int64),
 Product type                  object
 SKU                           object
 Price                        float64
 Availability                   int64
 Number of products sold        int64
 Revenue generated            float64
 Customer demographics         object
 Stock levels                   int64
 Lead times                     int64
 Order quantities               int64
 Shipping times                 int64
 Shipping carriers             object
 Shipping costs               float64
 Supplier name                 object
 Location                      object
 Lead time                      int64
 Production volumes             int64
 Manufacturing lead time        int64
 Manufacturing costs          float64
 Inspection results            object
 Defect rates                 float64
 Transportation modes          object
 Routes                        object
 Costs                        float64
 dtype: object)
```

In [19]:
```python
supply_data.head()
```

Out[19]:

	Product type	SKU	Price	Availability	Number of products sold	Revenue generated	Customer demographics	Stock levels	Lead times	Order quantities
0	haircare	SKU0	69.808006	55	802	8661.996792	Non-binary	58	7	96
1	skincare	SKU1	14.843523	95	736	7460.900065	Female	53	30	37
2	haircare	SKU2	11.319683	34	8	9577.749626	Unknown	1	10	88
3	skincare	SKU3	61.163343	68	83	7766.836426	Non-binary	23	13	59
4	skincare	SKU4	4.805496	26	871	2686.505152	Non-binary	5	3	56

5 rows × 24 columns

In [35]:

```python
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, r2_score
import numpy as np
import seaborn as sns

# Define target and features
target = 'Revenue generated'
features = supply_data.drop(columns=[target, 'SKU'])  # Drop SKU as it's an ID-like column
X = features
y = supply_data[target]

# Identify categorical and numerical columns
categorical_cols = X.select_dtypes(include=['object']).columns.tolist()
numerical_cols = X.select_dtypes(exclude=['object']).columns.tolist()

# Build preprocessing and model pipeline
preprocessor = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols)
    ],
    remainder='passthrough'  # Keep other columns as-is
)

# Pipeline with Random Forest
model = Pipeline(steps=[
    ('preprocessor', preprocessor),
    ('regressor', RandomForestRegressor(random_state=42))
])

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

mse, r2
```

Out[35]:

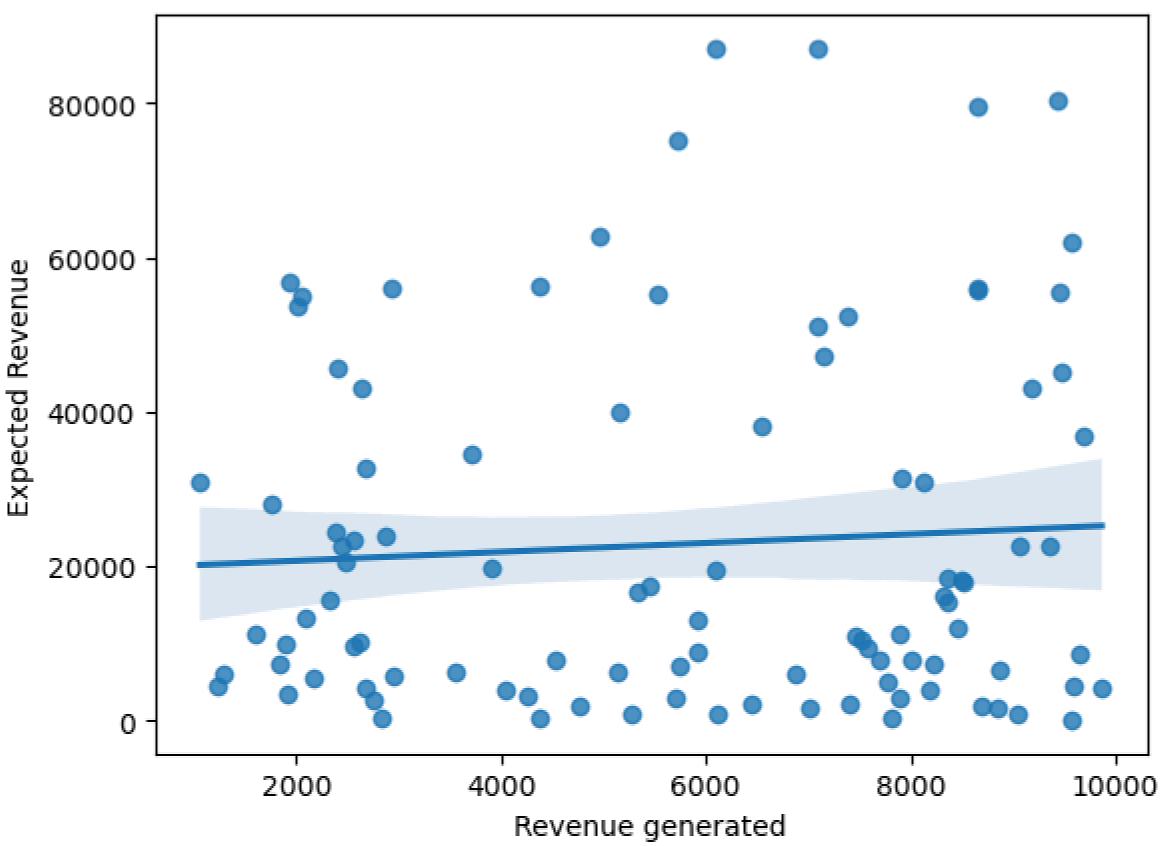
```
(9149773.40937217, -0.13858036544453345)
```

In [45]:

```python
import seaborn as sns
# Drop irrelevant or potentially redundant columns
df_cleaned = supply_data.drop(columns=['SKU', 'Lead time'])

# Optional feature engineering (check correlation later)
df_cleaned['Expected Revenue'] = df_cleaned['Price'] * df_cleaned['Number of products sold']

# Check correlation of 'Expected Revenue' with 'Revenue generated'
correlation = df_cleaned[['Expected Revenue', 'Revenue generated']].corr().iloc[0, 1]
sns.regplot(x='Revenue generated',y='Expected Revenue',data=df_cleaned)

# Define X and y
X_cleaned = df_cleaned.drop(columns=['Revenue generated'])
```

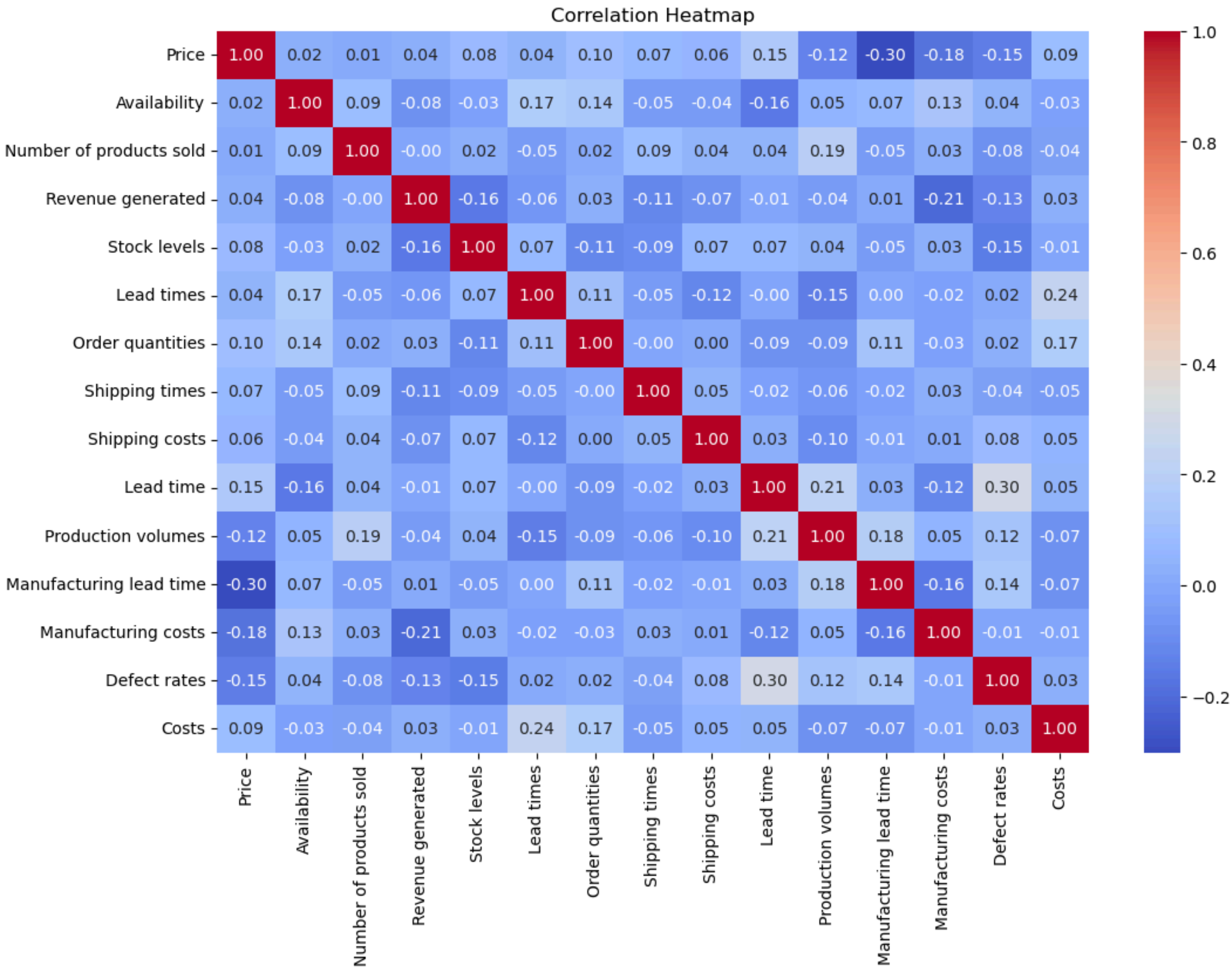
```python
y_cleaned = df_cleaned['Revenue generated']

# Identify new categorical and numerical columns
categorical_cols_cleaned = X_cleaned.select_dtypes(include=['object']).columns.tolist()
numerical_cols_cleaned = X_cleaned.select_dtypes(exclude=['object']).columns.tolist()

# Preprocessing and model pipeline
preprocessor_cleaned = ColumnTransformer(
    transformers=[
        ('cat', OneHotEncoder(handle_unknown='ignore'), categorical_cols_cleaned)
    ],
    remainder='passthrough'
)

model_cleaned = Pipeline(steps=[
    ('preprocessor', preprocessor_cleaned),
    ('regressor', RandomForestRegressor(random_state=42))
])

# Train/test split
X_train_clean, X_test_clean, y_train_clean, y_test_clean = train_test_split(
    X_cleaned, y_cleaned, test_size=0.2, random_state=42)

# Fit model
model_cleaned.fit(X_train_clean, y_train_clean)

# Evaluate
y_pred_clean = model_cleaned.predict(X_test_clean)
mse_clean = mean_squared_error(y_test_clean, y_pred_clean)
r2_clean = r2_score(y_test_clean, y_pred_clean)

correlation, mse_clean, r2_clean
```

Out[45]: (0.06878818868423413, 8495410.975455567, -0.0571527512504173)



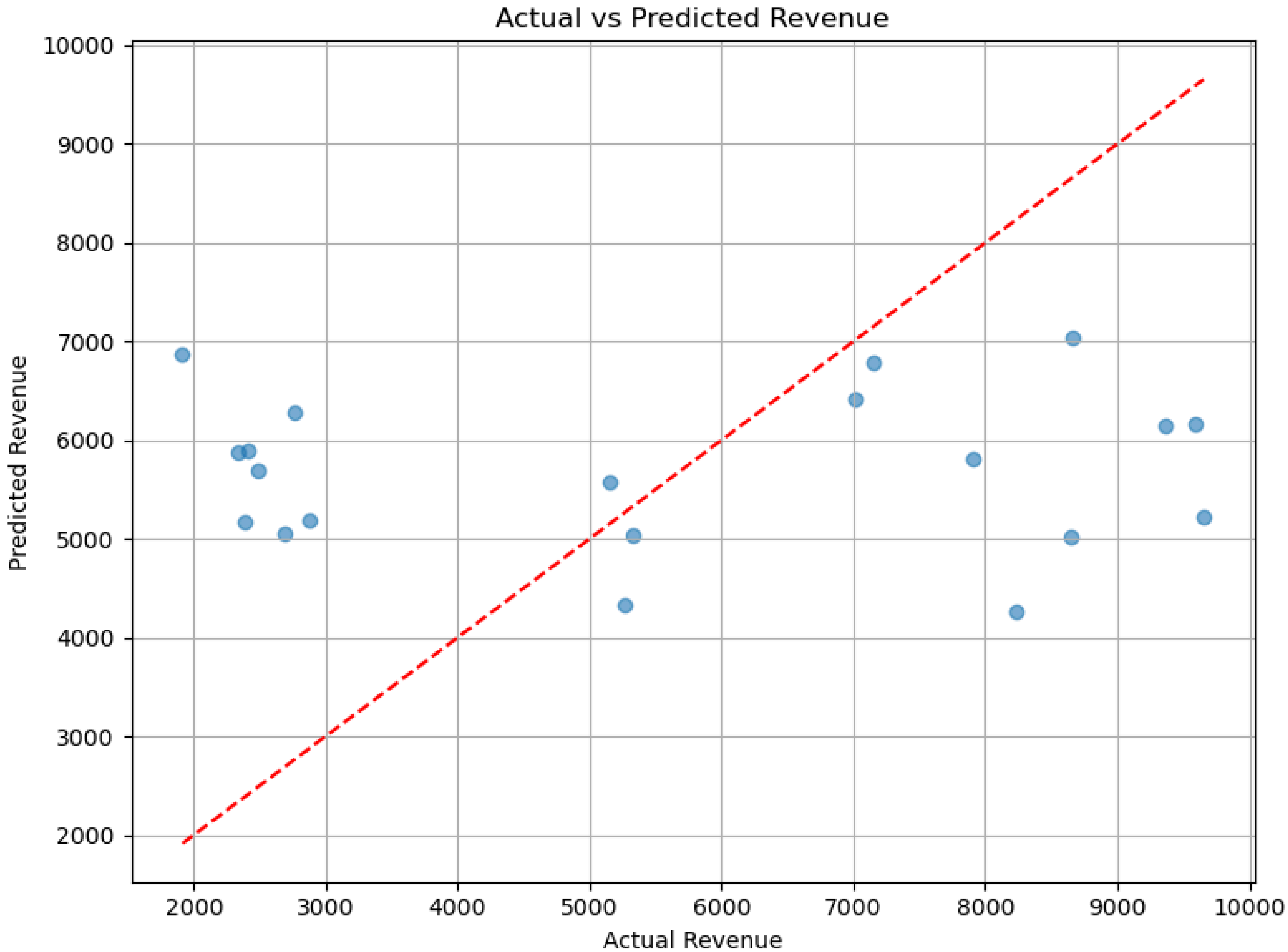
In [53]:

```python
import seaborn as sns

# Only numerical features
numerical_df = supply_data.select_dtypes(exclude=['object'])

plt.figure(figsize=(12, 8))
sns.heatmap(numerical_df.corr(), annot=True, fmt=".2f", cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```

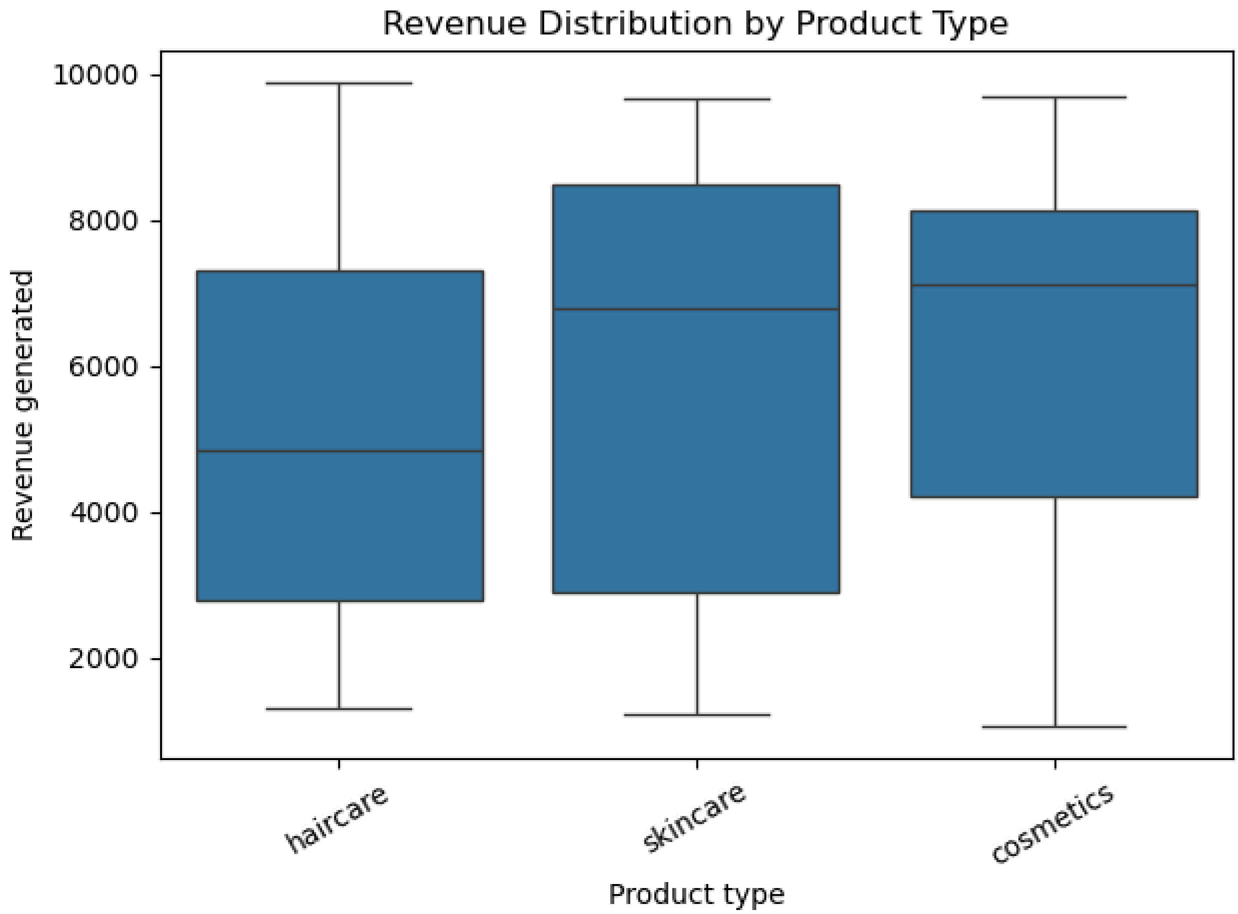


In [55]:

```python
import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
plt.scatter(y_test_clean, y_pred_clean, alpha=0.6)
plt.plot([y_test_clean.min(), y_test_clean.max()],
         [y_test_clean.min(), y_test_clean.max()],
         'r--')
plt.xlabel("Actual Revenue")
plt.ylabel("Predicted Revenue")
plt.title("Actual vs Predicted Revenue")
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [61]:
```
sns.boxplot(x='Product type', y='Revenue generated', data=supply_data)
plt.title("Revenue Distribution by Product Type")
plt.xticks(rotation=30)
plt.tight_layout()
plt.show()
```


Revenue Distribution by Product Type
10000
8000
6000
4000
2000
Revenue generated
haircare
skincare
cosmetics
Product type

```
In [ ]:
```