```
In [5]:  # STEP 1: Install Libraries
          print("STEP 1: Installing required libraries...")
          !pip install nltk scikit-learn pandas --quiet
```

```
STEP 1: Installing required libraries...
```

```
In [9]:  # STEP 2: Import Libraries
          print("STEP 2: Importing libraries...")
          import pandas as pd
          import nltk
          import string
          from nltk.tokenize import word_tokenize
          from nltk.corpus import stopwords
          from sklearn.feature_extraction.text import TfidfVectorizer
          from sklearn.metrics.pairwise import cosine_similarity
```

```
STEP 2: Importing libraries...
```

```
In [11]:  # STEP 3: Download NLTK Resources
          print("STEP 3: Downloading punkt and stopwords...")
          nltk.download('punkt')
          nltk.download('stopwords')
```

```
STEP 3: Downloading punkt and stopwords...
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Tcs\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Tcs\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
Out[11]:  True
```

```
In [15]:  # STEP 4: Load Dataset

          data = {
              'Text': [
                  "Absolutely wonderful - silky and sexy and comfortable.",
                  "Love this dress! it's sooo pretty.",
                  "I had to return it - the fit was just not right.",
                  "Terrible quality. Do not recommend.",
                  "Fast shipping and good packaging, but the product is bad.",
                  "The color is not the same as shown in the picture."
              ]
          }
          df = pd.DataFrame(data)

          # 4a: Select 'Text' column
          print("\n4a: Selected column 'Text'")
          print(df['Text'].head())

          # 4b: Remove nulls
          print("\n4b: Removing missing/null values")
          df.dropna(subset=['Text'], inplace=True)
```

```
# 4c: Keep first 10,000 records
print("\n4c: Keeping top 10,000 reviews (if present)")
df = df.head(10000)
print(df)
```

```
4a: Selected column 'Text'
0    Absolutely wonderful - silky and sexy and comf...
1                       Love this dress! it's sooo pretty.
2     I had to return it - the fit was just not right.
3                     Terrible quality. Do not recommend.
4    Fast shipping and good packaging, but the prod...
Name: Text, dtype: object

4b: Removing missing/null values

4c: Keeping top 10,000 reviews (if present)
                                                Text
0  Absolutely wonderful - silky and sexy and comf...
1                     Love this dress! it's sooo pretty.
2   I had to return it - the fit was just not right.
3                   Terrible quality. Do not recommend.
4  Fast shipping and good packaging, but the prod...
5  The color is not the same as shown in the pict...
```

In [27]:
```
# STEP 5: Load Stopwords
stop_words = set(stopwords.words('english'))
print("5a: Number of stopwords loaded:", len(stop_words))
```

```
5a: Number of stopwords loaded: 198
```

In [39]:
```
import nltk
nltk.download('punkt', quiet=False)
from nltk.tokenize import word_tokenize
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Tcs\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

In [41]:
```
import nltk
nltk.download('punkt')
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

stop_words = set(stopwords.words('english'))

def preprocess_text(text):
    print("Original:", text)
    text = text.lower()
    print("Lower:", text)
    text = ''.join([c for c in text if c.isalnum() or c.isspace()])
    print("No punctuation:", text)
    tokens = word_tokenize(text)
    print("Tokenized:", tokens)
    tokens = [w for w in tokens if w not in stop_words]
    print("No stopwords:", tokens)
    return ' '.join(tokens)

print(preprocess_text("This dress is really pretty!"))
```

```
Original: This dress is really pretty!
Lower: this dress is really pretty!
No punctuation: this dress is really pretty
Tokenized: ['this', 'dress', 'is', 'really', 'pretty']
No stopwords: ['dress', 'really', 'pretty']
dress really pretty
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\Tcs\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\Tcs\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

In [45]:

```
# STEP 7: Apply Preprocessing
df['cleaned'] = df['Text'].apply(preprocess_text)
```

```
Original: Absolutely wonderful - silky and sexy and comfortable.
Lower: absolutely wonderful - silky and sexy and comfortable.
No punctuation: absolutely wonderful  silky and sexy and comfortable
Tokenized: ['absolutely', 'wonderful', 'silky', 'and', 'sexy', 'and', 'comforta
ble']
No stopwords: ['absolutely', 'wonderful', 'silky', 'sexy', 'comfortable']
Original: Love this dress! it's sooo pretty.
Lower: love this dress! it's sooo pretty.
No punctuation: love this dress its sooo pretty
Tokenized: ['love', 'this', 'dress', 'its', 'sooo', 'pretty']
No stopwords: ['love', 'dress', 'sooo', 'pretty']
Original: I had to return it - the fit was just not right.
Lower: i had to return it - the fit was just not right.
No punctuation: i had to return it  the fit was just not right
Tokenized: ['i', 'had', 'to', 'return', 'it', 'the', 'fit', 'was', 'just', 'no
t', 'right']
No stopwords: ['return', 'fit', 'right']
Original: Terrible quality. Do not recommend.
Lower: terrible quality. do not recommend.
No punctuation: terrible quality do not recommend
Tokenized: ['terrible', 'quality', 'do', 'not', 'recommend']
No stopwords: ['terrible', 'quality', 'recommend']
Original: Fast shipping and good packaging, but the product is bad.
Lower: fast shipping and good packaging, but the product is bad.
No punctuation: fast shipping and good packaging but the product is bad
Tokenized: ['fast', 'shipping', 'and', 'good', 'packaging', 'but', 'the', 'prod
uct', 'is', 'bad']
No stopwords: ['fast', 'shipping', 'good', 'packaging', 'product', 'bad']
Original: The color is not the same as shown in the picture.
Lower: the color is not the same as shown in the picture.
No punctuation: the color is not the same as shown in the picture
Tokenized: ['the', 'color', 'is', 'not', 'the', 'same', 'as', 'shown', 'in', 't
he', 'picture']
No stopwords: ['color', 'shown', 'picture']
```

In [47]:
```python
# STEP 8: TF-IDF Vectorization
vectorizer = TfidfVectorizer()
tfidf_matrix = vectorizer.fit_transform(df['cleaned'])
print("8a: TF-IDF Matrix Shape ->", tfidf_matrix.shape)
```

```
8a: TF-IDF Matrix Shape -> (6, 24)
```

In [59]:
```python
import numpy as np
from sklearn.metrics.pairwise import cosine_similarity

# Function to preprocess input query just like reviews
def preprocess_text(text):
    # Lowercase
    text = text.lower()
    # Remove punctuation and non-alphabetic characters
    text = re.sub(r'[^a-z\s]', '', text)
    # Tokenize
    tokens = nltk.word_tokenize(text)
    # Remove stopwords
    tokens = [t for t in tokens if t not in stopwords.words('english')]
    # Lemmatize
    tokens = [lemmatizer.lemmatize(t) for t in tokens]
    return ' '.join(tokens)

# Step 9a to 9e: Retrieval function
def retrieve_top_k_reviews(query, k=3):
    cleaned_query = preprocess_text(query)
    query_vector = vectorizer.transform([cleaned_query])  # Step 9b
    similarities = cosine_similarity(query_vector, tfidf_matrix).flatten()  #
    top_k_indices = similarities.argsort()[-k:][::-1]  # Step 9d
    print(f"\nQuery: {query}")
    print("\nTop matching reviews:\n")
    for idx in top_k_indices:
        print(f"Original Review: {df.iloc[idx]['Text']}")
        print(f"Cleaned Review: {df.iloc[idx]['cleaned']}")
        print(f"Similarity Score: {similarities[idx]:.4f}")
        print("-" * 60)
```

```
In [ ]:
```