Welcome to the world of e-commerce, where customer feedback is a goldmine of insights! In this project, you'll dive into the Women's Clothing E-Commerce Reviews dataset, focusing on the 'Review Text' column filled with direct customer opinions.

Your mission is to use text embeddings and Python to analyze these reviews, uncover underlying themes, and understand customer sentiments. This analysis will help improve customer service and product offerings.

The Data

You will be working with a dataset specifically focusing on customer reviews. Below is the data dictionary for the relevant field:

womens_clothing_e-commerce_reviews.csv

Column	Description
`'Review Text'`	Textual feedback provided by customers about their shopping experience and product quality.

Armed with access to powerful embedding API services, you will process the reviews, extract meaningful insights, and present your findings.

Let's get started!

Install useful libraries

```python
# Run this cell to install ChromaDB if desired
try:
    assert version('chromadb') == '0.4.17'
except:
    !pip install chromadb==0.4.17
try:
    assert version('pysqlite3') == '0.5.2'
except:
    !pip install pysqlite3-binary==0.5.2
__import__('pysqlite3')
import sys
sys.modules['sqlite3'] = sys.modules.pop('pysqlite3')
import chromadb
import os
import openai
import pandas as pd
import numpy as np
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
from scipy.spatial import distance
from chromadb.utils.embedding_functions import OpenAIEmbeddingFunction
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: chromadb==0.4.17 in /home/repl/.local/lib/python3.10/site-packages (0.4.17)
Requirement already satisfied: requests>=2.28 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (2.32.3)
Requirement already satisfied: pydantic>=1.9 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (2.7.1)
Requirement already satisfied: chroma-hnswlib==0.7.3 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (0.7.3)
Requirement already satisfied: fastapi>=0.95.2 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (0.115.8)
Requirement already satisfied: uvicorn>=0.18.3 in /usr/local/lib/python3.10/dist-packages (from uvicorn[standard]>=0.18.3-
>chromadb==0.4.17) (0.34.0)
Requirement already satisfied: posthog>=2.4.0 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (3.12.1)
Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17)
(4.12.2)
Requirement already satisfied: pulsar-client>=3.1.0 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (3.6.1)
Requirement already satisfied: onnxruntime>=1.14.1 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (1.20.1)
Requirement already satisfied: opentelemetry-api>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17)
(1.30.0)
Requirement already satisfied: opentelemetry-exporter-otlp-proto-grpc>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from
chromadb==0.4.17) (1.30.0)
Requirement already satisfied: opentelemetry-sdk>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17)
(1.30.0)
Requirement already satisfied: tokenizers>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (0.21.0)
Requirement already satisfied: pypika>=0.48.9 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (0.48.9)
Requirement already satisfied: tqdm>=4.65.0 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (4.67.1)
Requirement already satisfied: overrides>=7.3.1 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (7.7.0)
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (6.5.2)
Requirement already satisfied: grpcio>=1.58.0 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (1.70.0)
Requirement already satisfied: bcrypt>=4.0.1 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (4.2.1)
Requirement already satisfied: typer>=0.9.0 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (0.9.4)
Requirement already satisfied: kubernetes>=28.1.0 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (32.0.0)
Requirement already satisfied: tenacity>=8.2.3 in /usr/local/lib/python3.10/dist-packages (from chromadb==0.4.17) (8.5.0)
```

Load the dataset

Load data and perform basic data checks to ensure you are using relevant data for the analysis

```python
# Load the dataset
import pandas as pd
reviews = pd.read_csv("womens_clothing_e-commerce_reviews.csv")

# Display the first few entries
reviews.head()
```

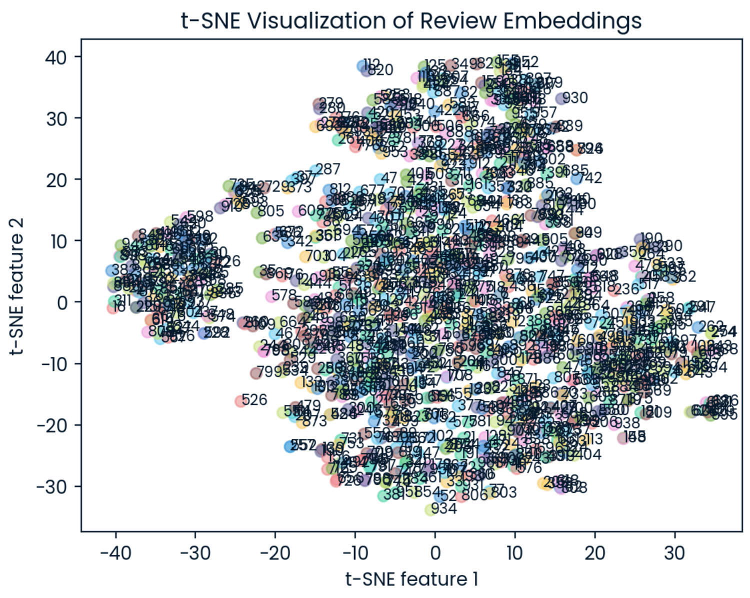
...	R	Clo...	...	Title	Review Text	...	Recomme...
0	0	767	33	null	Absolutely wonderful - silky and sexy and co...	4	
1	1	1080	34	null	Love this dress! it's sooo pretty. i happened t...	5	
2	2	1077	60	Some major design flaws	I had such high hopes for this dress and reall...	3	
3	3	1049	50	My favorite buy!	I love, love, love this jumpsuit. it's fun, flirty, a...	5	
4	4	847	47	Flattering shirt	This shirt is very flattering to all due to the a...	5	

Rows: 5 Expand

```python
# Start coding here
review_texts = reviews["Review Text"].dropna()

client = openai.OpenAI()
responses = client.embeddings.create(input=review_texts.tolist(), model="text-embedding-3-small").model_dump()
embeddings = [response["embedding"] for response in responses["data"]]

# Use as many cells as you need.
```

```python
def apply_tsne(embeddings):
    tsne = TSNE(n_components=2, random_state=0)
    return tsne.fit_transform(embeddings)

embeddings_2d = apply_tsne(np.array(embeddings))
```

```python
def plot_tsne(tsne_results):
    for i, point in enumerate(tsne_results):
        plt.scatter(point[0],point[1], alpha=0.5)
        plt.text(point[0], point[1], str(i), fontsize=8, verticalalignment='center')
    plt.title("t-SNE Visualization of Review Embeddings")
    plt.xlabel("t-SNE feature 1")
    plt.ylabel("t-SNE feature 2")
    plt.show()

plot_tsne(embeddings_2d)
```



```
categories = ['quality', 'fit', 'style', 'comfort', 'affordable', 'expensive']
category_responses = client.embeddings.create(input=categories, model="text-embedding-3-small").model_dump()
category_embeddings = [embedding['embedding'] for embedding in category_responses["data"]]
similarities = []

def category_feedback(text_embedding, category_embeddings):
        for i, cat_emb in enumerate(category_embeddings):
            dist = distance.cosine(text_embedding, cat_emb)
            similar = {"distance": dist, "index": i}
            similarities.append(similar)
        closest = min(similarities, key=lambda x: x["index"])
        return categories[closest["index"]]

feedback_categories = [category_feedback(embedding, category_embeddings) for embedding in embeddings]
```

```
import os
import uuid
import chromadb
from chromadb.utils.embedding_functions import OpenAIEmbeddingFunction

# Only create the collection if it doesn't exist, otherwise get it
db_client = chromadb.PersistentClient()
collection_name = "review_embeddings"
try:
    review_embeddings_db = db_client.create_collection(
        name=collection_name,
        embedding_function=OpenAIEmbeddingFunction(
            model_name="text-embedding-3-small",
            api_key=os.environ["OPENAI_API_KEY"]
        )
    )
    # Add documents only if collection was just created
    review_embeddings_db.add(
        documents=review_texts.tolist(),
        ids=[str(i) for i in range(len(review_texts))]
    )
except Exception as e:
    # If collection exists, just get it
    if "already exists" in str(e):
        review_embeddings_db = db_client.get_collection(
            name=collection_name,
            embedding_function=OpenAIEmbeddingFunction(
                model_name="text-embedding-3-small",
                api_key=os.environ["OPENAI_API_KEY"]
            )
        )
    else:
        raise

def find_similar_reviews(input_text, vector_db, n=3):
    results = vector_db.query(
        query_texts=[input_text],
        n_results=n
    )
    return results

example_review = "Absolutely wonderful - silky and sexy and comfortable"
most_similar_reviews = find_similar_reviews(example_review, review_embeddings_db, 3)["documents"][0]
print(most_similar_reviews)
```

```
['Absolutely wonderful - silky and sexy and comfortable', 'Very comfortable and versatile. got lots of compliments.', 'This is a
very comfortable and sexy sleep dress, the way it drapes. i can see that the type of fabric is not suitable for out and about
activities and can catch on snags, etc. very easily. once i relegated it to the nightgown category it became my favorite item to
wear and lounge in.']
```