# Project Report: Retrieval-Augmented Generation (RAG) on Myntra Fashion Data

## 1. Introduction

The project implements a Retrieval-Augmented Generation (RAG) system tailored for fashion product recommendations using Myntra’s fashion dataset. The goal is to combine vector similarity search with generative response capabilities, enabling natural language queries like 'Red floral kurta' to retrieve relevant products and provide contextual answers.

## 2. Dataset

Source: Myntra Fashion Product Dataset (Kaggle)  
Size: ~14,000 products  
  
Features Used:  
- name (product title)  
- brand  
- colour  
- price  
- img (image URL)  
- p\_id (product ID)  
- products (category/text description)  
  
Data cleaning steps included handling null values, normalizing text fields, and creating a 'text' field for embeddings.

## 3. System Architecture

### 3.1 Components

1. Embedding Layer  
 - Model: all-MiniLM-L6-v2 (SentenceTransformers)  
 - Purpose: Convert product text into dense vector embeddings.  
  
2. Vector Store  
 - Database: ChromaDB (in-memory)  
 - Collection: 'myntra\_gen\_search'  
 - Function: Store embeddings + metadata for fast retrieval.  
 - Batch insertion was implemented to avoid max batch size errors.  
  
3. Search Layer  
 - Uses cosine similarity over embeddings.  
 - Returns top-k matches (n\_results).  
 - Supports reranking for better relevance.  
  
4. Generation Layer  
 - Combines retrieved product metadata with generative prompts.  
 - Produces user-friendly recommendations in natural language.

## 4. Implementation Highlights

- Batch-safe indexing: Limited to 1,000 items per add operation to handle ChromaDB constraints.  
- Recommendation function:  
  
def recommend(query, n=5):  
 results = collection.query(query\_texts=[query], n\_results=n)  
 ...  
 return pd.DataFrame(recs)  
  
- Image rendering in Jupyter: Display product images in DataFrame outputs via embedded HTML.  
- Reranked results: Extended metadata returned with similarity score.

## 5. Example Workflow

- Query: 'Red floral kurta'  
- Retrieval: Top 5 kurta products with red/printed/floral attributes.  
- Display: Tabular output including product name, brand, colour, price, and image preview.  
- Enhanced Output: Reranked JSON with fields:  
  
{  
 "name": "...",  
 "brand": "...",  
 "colour": "...",  
 "price": "...",  
 "img": "...",  
 "score": 0.8943  
}

## 6. Evaluation

- Qualitative evaluation: Visual inspection of retrieved product sets.  
- Planned RAG Evaluation: Adding metrics such as:  
 - Retrieval precision/recall  
 - Embedding similarity distribution  
 - Human evaluation of recommendation relevance

## 7. Applications

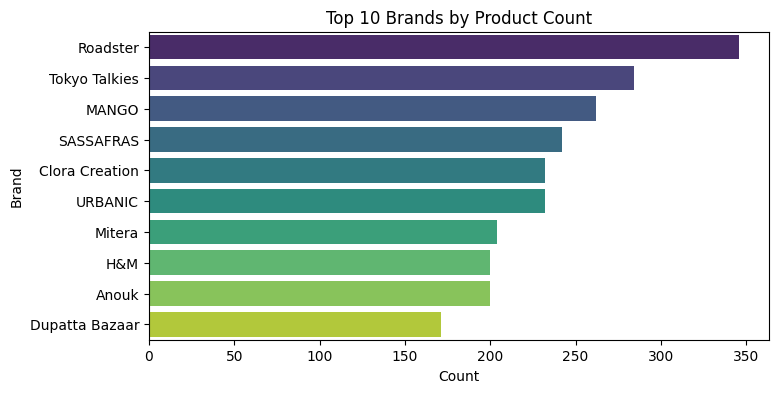
- Personalized fashion search engine.  
- Conversational shopping assistants.  
- Cross-selling & up-selling in e-commerce.

## 8. Future Work

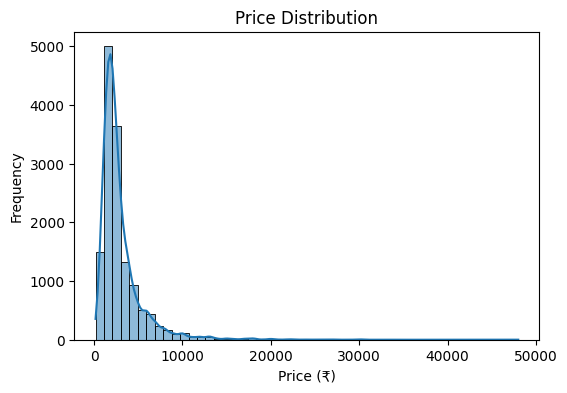
- Integrate cross-encoder rerankers (e.g., MiniLM cross-encoder).  
- Add LLM-based summarization of product sets (e.g., 'Here are 5 red kurtas under ₹1000').  
- Deploy as interactive web app using Streamlit/Gradio.  
- Experiment with larger embedding models (e.g., all-mpnet-base-v2) for better semantic recall.  
- Evaluate RAG metrics systematically.

# Appendix A: Notebook Figures & Plots

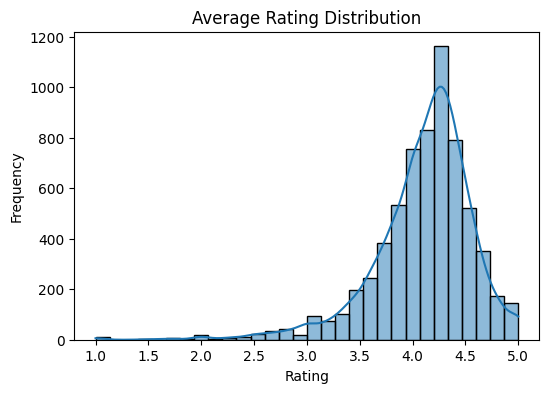
**Figure 1. Output image from code cell #9**



**Figure 2. Output image from code cell #10**

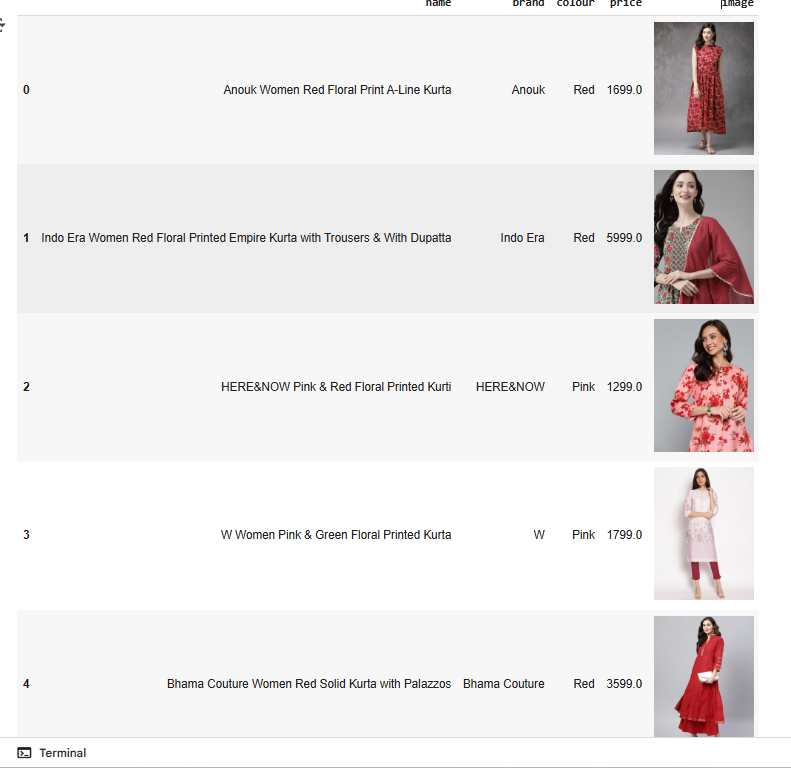


**Figure 3. Output image from code cell #11.**



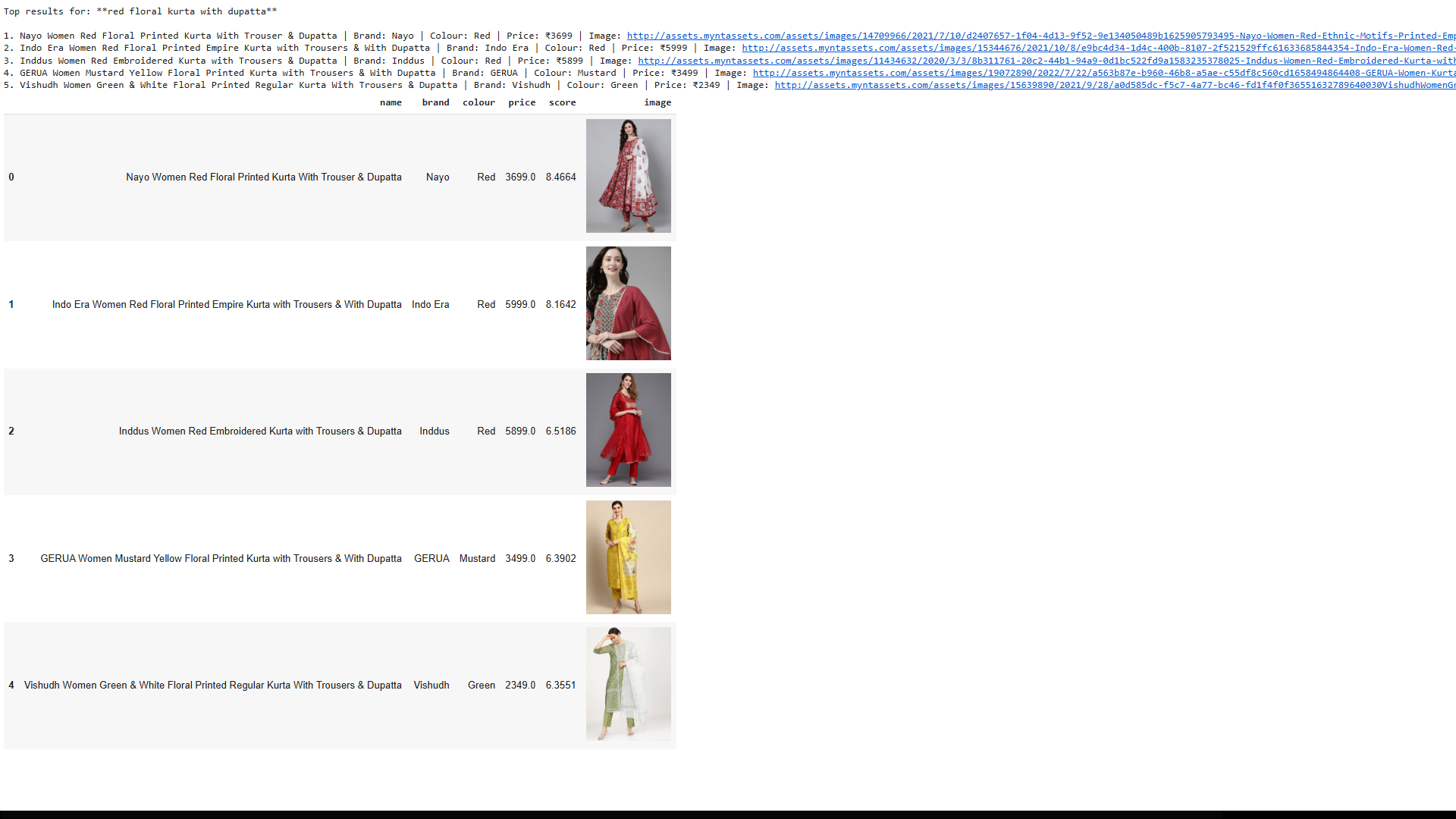
**Figure 4. Output image from code cell #23**

**Search Result for Red Floral Kurta**



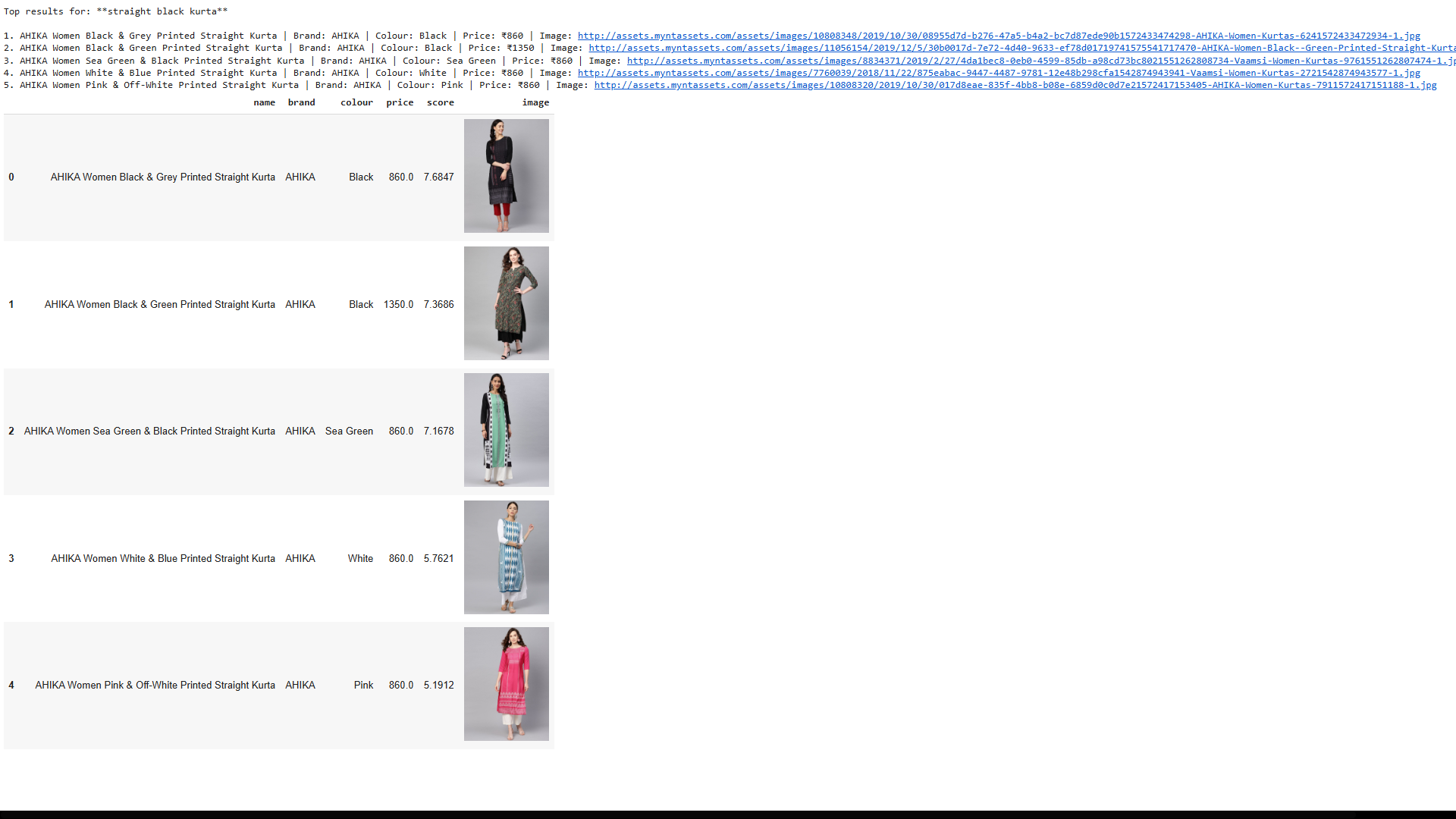
**Figure 5. Output image from code cell #34**

**Search Result for Red Floral Kurta with Dupatta (**Semantic Search) :



**Figure 6. Output image from code cell #35**

**Search Result for Straight Black Kurta with Filter of Brand = Ahika, Max Price = 2000**:



**Figure 7. Output image from code cell #36**

**Search Result for Straight Black Kurta with filters** **colour="Navy", price\_min=1000, price\_max=3000**

