FASHION SEARCH AI – LAMMAINDEX

AUTHOR: SHRINIVAS NANDKISHOR DHAMONE

1. INTRODUCTION

1.1 PROJECT GOALS

The Fashion Search AI project aims to create a generative search system that allows organizations to efficiently discover and validate fashion items, trends, and recommendations from a comprehensive fashion data repository. The system is designed to:

1. **Boost Productivity:** Cut down on the time required to search through fashion datasets by delivering precise and relevant search results.
2. **Aid Decision-Making:** Enable users to swiftly access both historical and current fashion trends to guide their choices and designs.
3. **Enhance Data Accessibility:** Make valuable information within the fashion dataset readily accessible and actionable.
4. **Improve Efficiency:** Optimize the process of retrieving fashion-related information, leading to better resource utilization and more informed decision-making in the fashion industry.

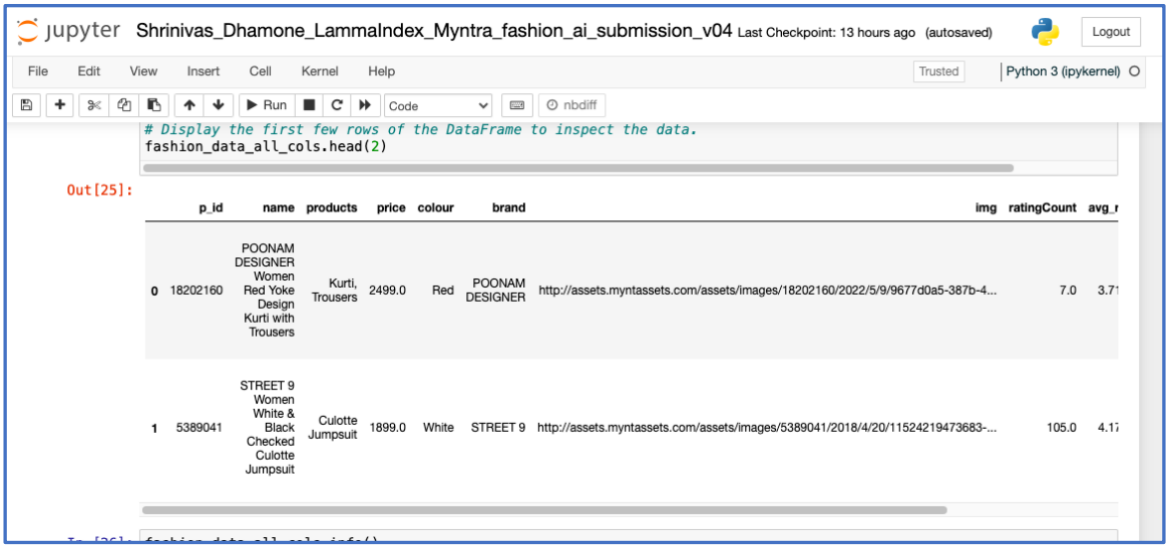
1.2 SCOPE

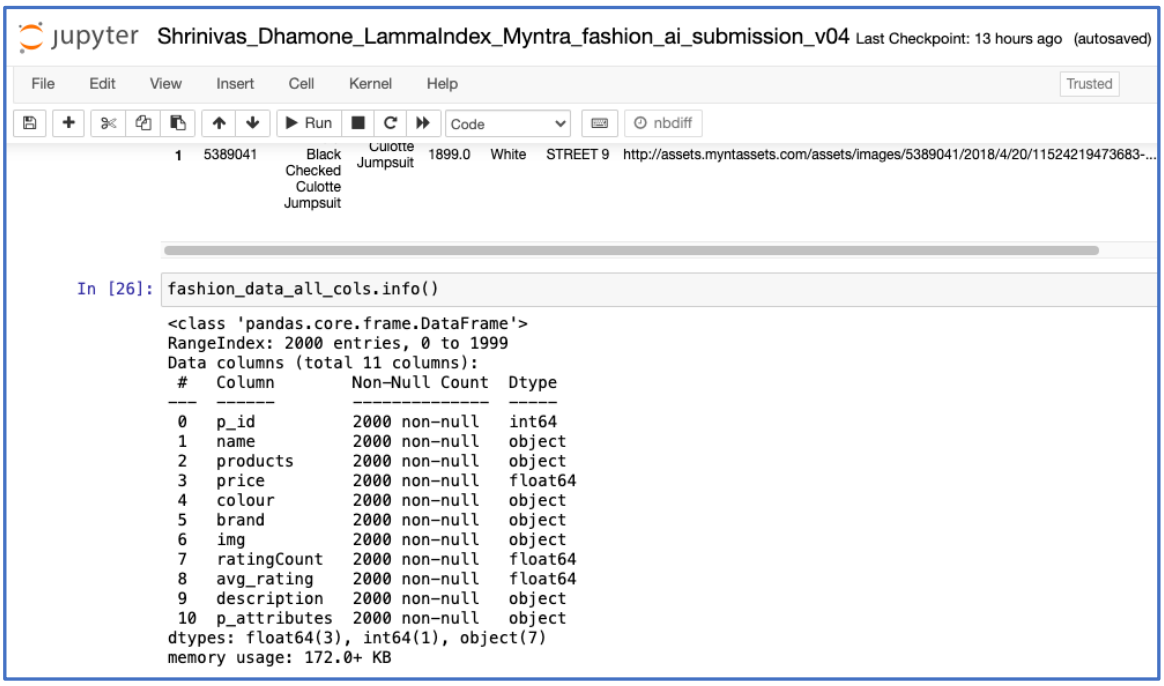
Objective

To develop a sophisticated search and information retrieval system tailored for fashion-related content by leveraging fashion data from Kaggle. This system will enhance users' ability to quickly and efficiently find relevant fashion information.

Components

1. **Data Ingestion**
    - **Objective:** Import and pre-process fashion-related data from Kaggle datasets.
    - **Details:**
        - Source data includes product descriptions, images, user reviews, and fashion trends.
        - Tasks involve cleaning, normalizing, and transforming this data into a structured format suitable for analysis and integration into the search system.





2. **Data Parsing**
    - **Objective:** Extract and organize relevant information from fashion-related datasets.
    - **Details:**
        - Metadata extraction focusing on fashion-related information.
        - Apply natural language processing (NLP) to identify and tag fashion-related topics and queries within the content.
3. **Embedding Layer**
    - **Objective:** Create embeddings for fashion content to enable semantic understanding and effective search.
    - **Details:**
        - Use pre-trained language models (e.g., BERT, GPT) to convert fashion content into high-dimensional vectors.
        - Generate embeddings from the provided dataset to create a unified representation.
        - Ensure embeddings capture the semantic nuances relevant to fashion.
4. **Search & Rank Layer**
    - **Objective:** Develop algorithms to search and rank fashion data based on user query relevance.

- **Details:**
  - Implement search algorithms (e.g., cosine similarity, BM25) to match query embeddings with fashion data embeddings.
  - Design ranking mechanisms to prioritize results based on relevance scores, incorporating factors such as query context and user preferences.
  - Optimize for fast response times and accurate results.

5. **Generation Layer**
   - **Objective:** Generate responses and identify similarities from the retrieved fashion content.
   - **Details:**
     - Use NLP techniques to create concise summaries.
     - Implement text generation models to provide direct answers to fashion-related questions.
     - Ensure that generated responses are contextually accurate and useful for users.

Expected Outcomes

- **Enhanced Search Efficiency:** Users will locate relevant fashion information from the dataset and Kaggle data more quickly and accurately.
- **Semantic Understanding:** The system will effectively interpret and respond to fashion-related queries with high relevance, leveraging embeddings and advanced search algorithms.
- **User Satisfaction:** By generating clear and useful responses, users will more effectively find the information they need, leading to increased satisfaction.

Success Metrics

- **Search Accuracy:** Precision and recall of search results in relation to user queries.
- **Response Time:** Speed of retrieval and generation of relevant information.
- **User Feedback:** Qualitative feedback from users regarding the relevance and usefulness of the search results and generated content.

Out of Scope

- **User Interface:** Designing a user-friendly interface for fashion search queries and displaying search results.
- **Security and Privacy:** Addressing data security and compliance with privacy regulations.

2. AI MODEL

2.1 FRAMEWORK SELECTION

SUITABILITY OF LLAMA INDEX FOR THE FASHION SEARCH AI PROJECT

Llama Index (formerly known as GPT Index) is highly suited for the Fashion Search AI project due to its advanced natural language processing (NLP) capabilities, ease of integration, and scalability. Here's why LlamaIndex stands out:

- **Contextual Understanding:** Llama Index excels in maintaining context across larger content data set , which is crucial for parsing and accurately retrieving information from sequences of fashion-related exchanges. This capability ensures precise understanding and validation of detailed information.
- **Easy Integration:** With robust APIs and integration tools, Llama Index can be seamlessly incorporated into existing systems, facilitating smooth implementation without significant redevelopment.

- **Semantic Search Capabilities:** Llama Index supports advanced semantic search, going beyond mere keyword matching. This feature enables the system to comprehend the meaning behind fashion-related queries and retrieve response based on the underlying content, leading to more relevant and accurate search results.
- **Scalability:** Designed to accommodate growing data volumes, Llama Index efficiently handles increasing amounts of fashion-related data without compromising performance, ensuring long-term effectiveness as the dataset expands.

These attributes make Llama Index an excellent choice for developing a comprehensive and efficient fashion search solution.

2.2 LLM MODEL SELECTION

- **LLM Model:** GPT-3.5-turbo by OpenAI
- **Embedding Model:** text-embedding-ada-002

2.3 PARSER

- **RecursiveCharacterTextSplitter:** This tool breaks down text into smaller segments recursively, ensuring that each chunk adheres to a specified character limit. It helps manage large texts while preserving context.
- **LangchainNodeParser:** This parser processes the smaller text chunks to extract meaningful information, maintain context, and prepare the data for further analysis or search operations.

2.4 MODEL TRAINING

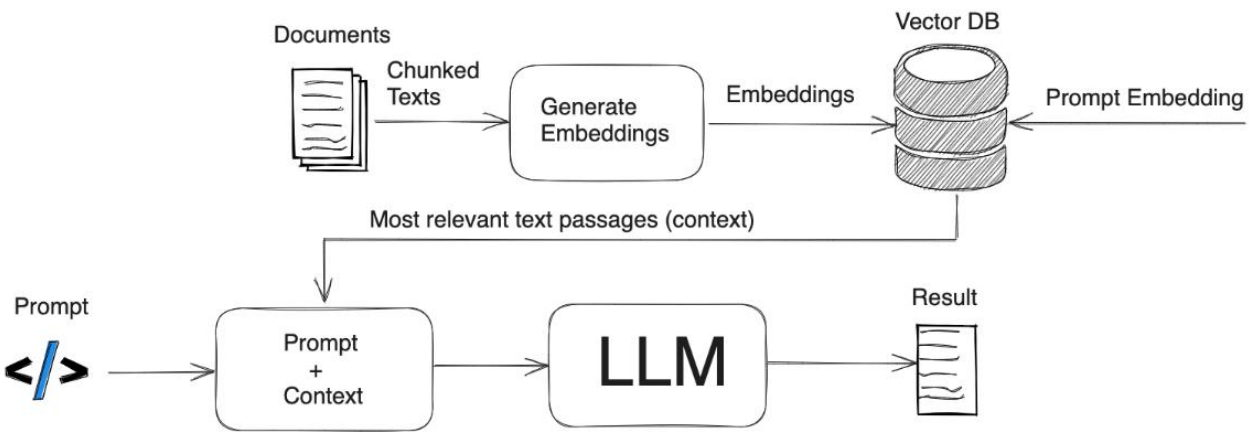
- **Initial Dataset:** The model was pre-trained using fashion data collected from Kaggle. You can access the dataset:

  https://www.kaggle.com/datasets/hiteshsuthar101/myntra-fashion-product-dataset

- **Fine-Tuning Data:** We developed detailed and elaborated prompts to guide the LLM in generating responses based on the context retrieved from the Vector DB.

3. KEY DESIGN STAGES

Implementing RAG model using LlamaIndex framework



3.1 PART 1 – DATA INGESTION

The embedding layer is responsible for transforming fashion related content into dense vector representations that capture the semantic meaning of the text. This layer involves:

- **Data loading**: Loading data from csv file using CSVReader.
- **Preprocessing**: Cleaning and normalizing the data by cleansing HTML related tags.
- **Tokenization**: Breaking down the text into chunks using RecursiveCharacterTextSplitter that can be processed by machine learning models.
- **Embedding chunks**: Using pre-trained language models (text-embedding-ada-002) to generate embeddings for each fashion related query
- **Embedding Storage**: Storing the embeddings in a Chroma database for efficient retrieval.

3.2 PART 2 – DATA RETRIEVAL

The search layer handles the retrieval of relevant fashion recommendation according to their relevance to the user query. This layer includes:

- **Query Embedding**: Transforming the user query into an embedding using the same model as in the embedding layer.
- **Similarity Search in vector DB:** Search to find similar fashion trends whose embeddings are close to the query embedding in vector DB. The extracted result is then added to cache for future reference.

3.3 PART 3 - GENERATION LAYER

The generation layer is responsible for creating summaries or direct answers based on the retrieved fashion data related content. This layer consists of:

- **Contextual Understanding**: Analysing the context of the retrieved fashion data to generate coherent and contextually accurate responses.
- **Response Generation**: Using generative models (OpenAI API) to produce summaries or answers. Prompt templates used to give proper structure to the response.
- **Post-Processing**: Refining the generated response to ensure clarity, accuracy, and relevance.

4. INNOVATION AND CREATIVITY

- **CSVReader** –
  Using CSVReader is advantageous when working with structured data in CSV files due to its ability to handle field-level access, custom parsing logic, metadata extraction, efficient data loading, and preprocessing capabilities. This makes it a powerful tool for building more sophisticated and efficient retrieval-augmented generation systems with structured datasets.
- **RecursiveCharacterTextSplitter** –
  It is ideal for applications that need to process text in chunks while maintaining semantic integrity and handling various text structures.
- **text-embedding-ada-002 model**: It is a versatile tool that can enhance various NLP applications by providing rich and meaningful representations of text data.
- **ChromaDB** –
  While VectorStoreIndex provides basic indexing functionality, ChromaVectorStore offers significant enhancements in terms of performance, scalability, and advanced search capabilities. For an Fashion Search AI project, these benefits

translate into faster, more accurate, and more efficient search and retrieval of relevant response, making ChromaVectorStore a valuable component in building a robust and high-performing search system.

5. CHALLENGES ENCOUNTERED

Case 1:

- **Issue:** The CSV file was read as if it were a document file, which was excessively large. This mishandling resulted in inaccurate context capture of the fashion dataset, causing delays in execution.
- **Solution:** We addressed this by working with a reduced dataset of 2,000 rows due to infrastructure constraints in Google Colab.

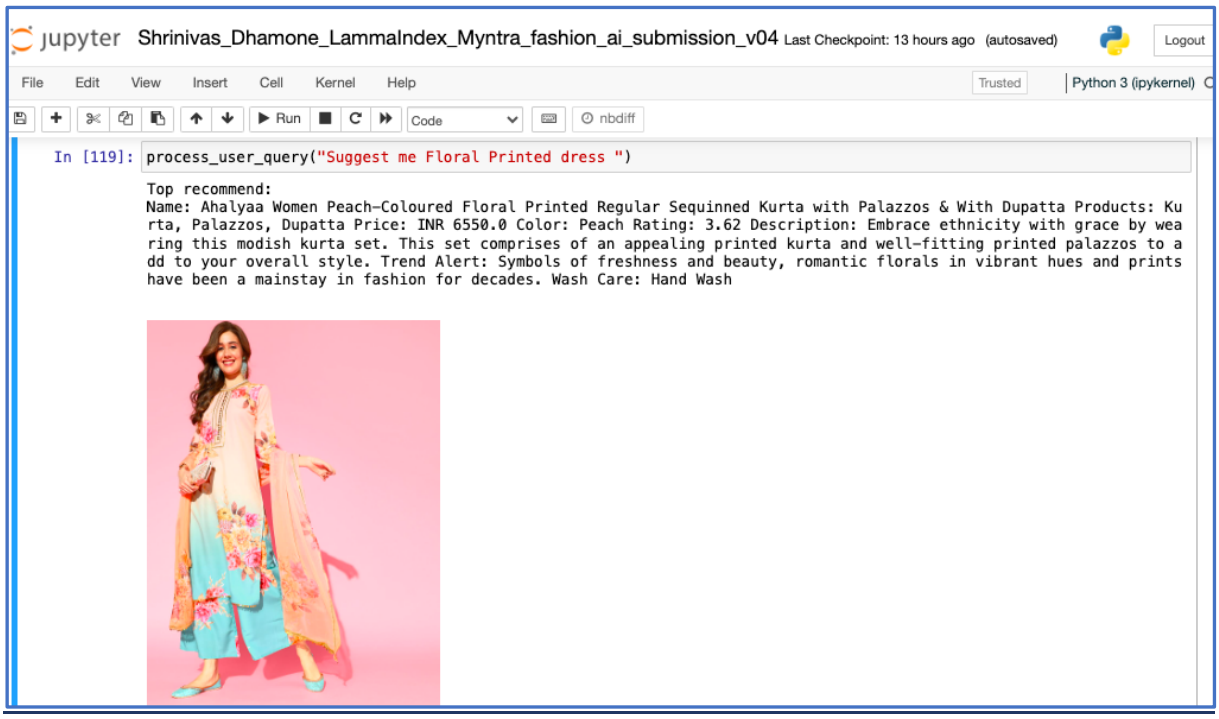
Case 2:

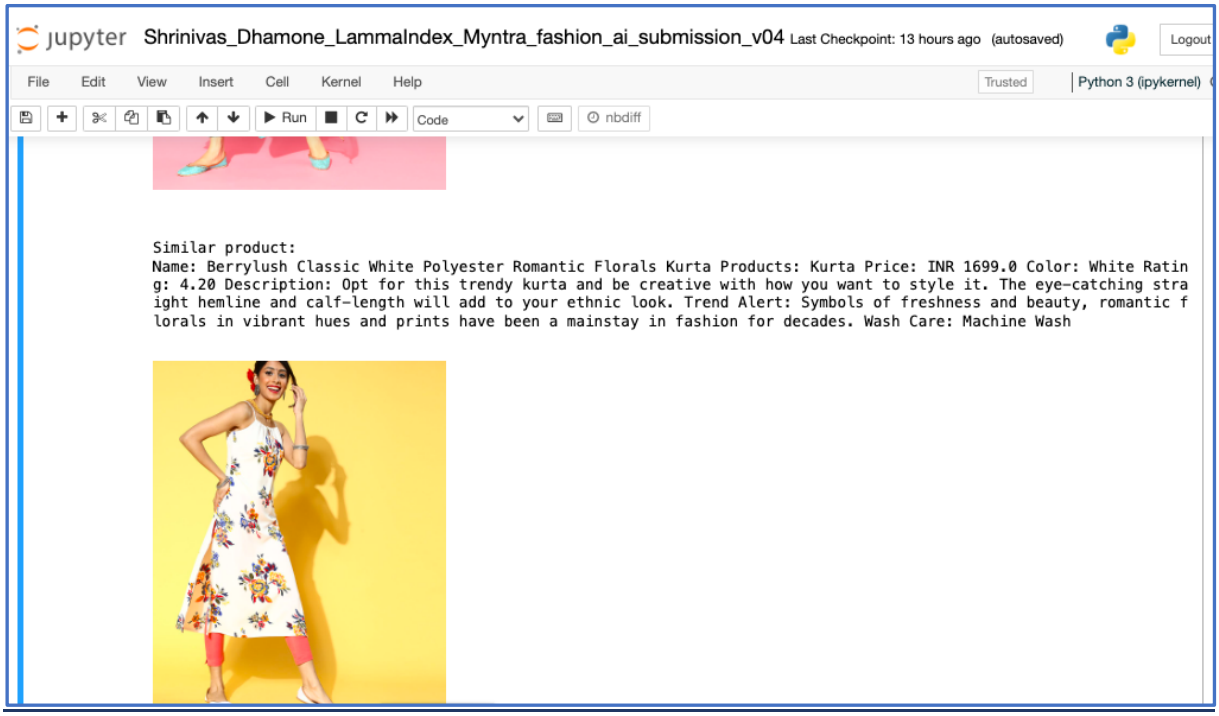
- **Issue:** Limited library dependencies on local machine installations led to difficulties, especially with the chroma vector store integration using Llama Index.
- **Solution:** We relied on libraries available in Google Colab, though this approach introduced various challenges during both coding and execution.

6. RESULTS / OUTPUT

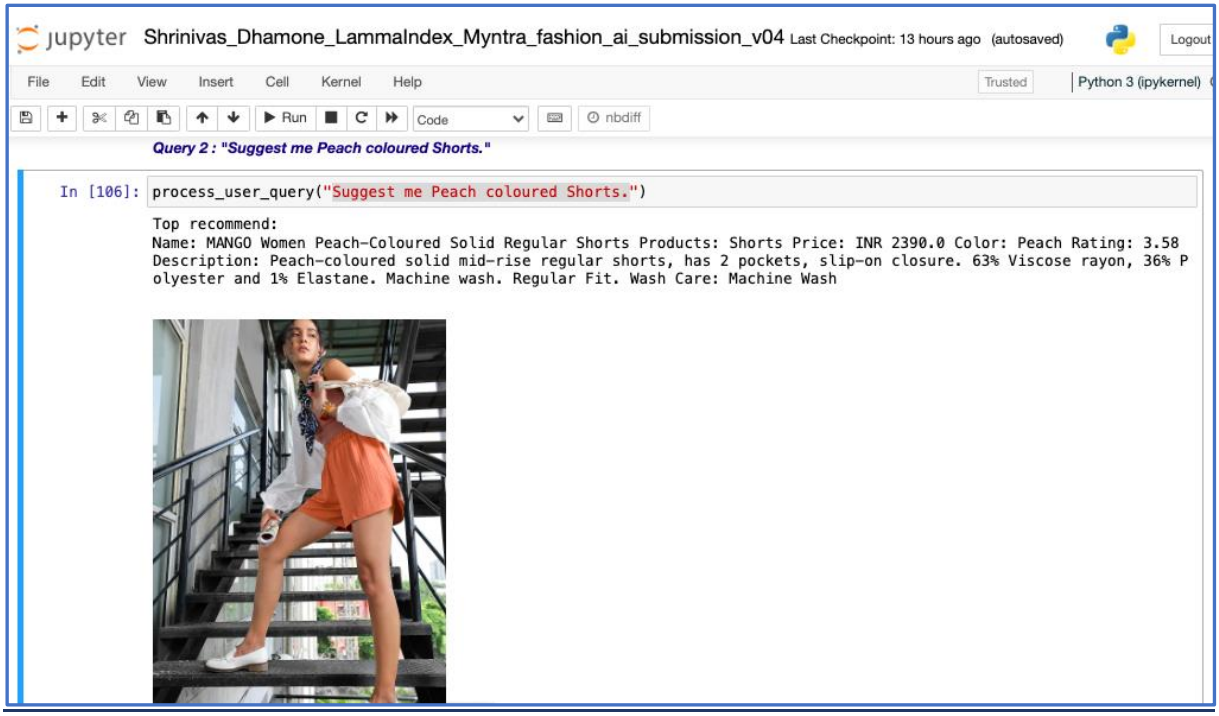
Below are screenshots showing the questions asked and the responses generated by the bot.

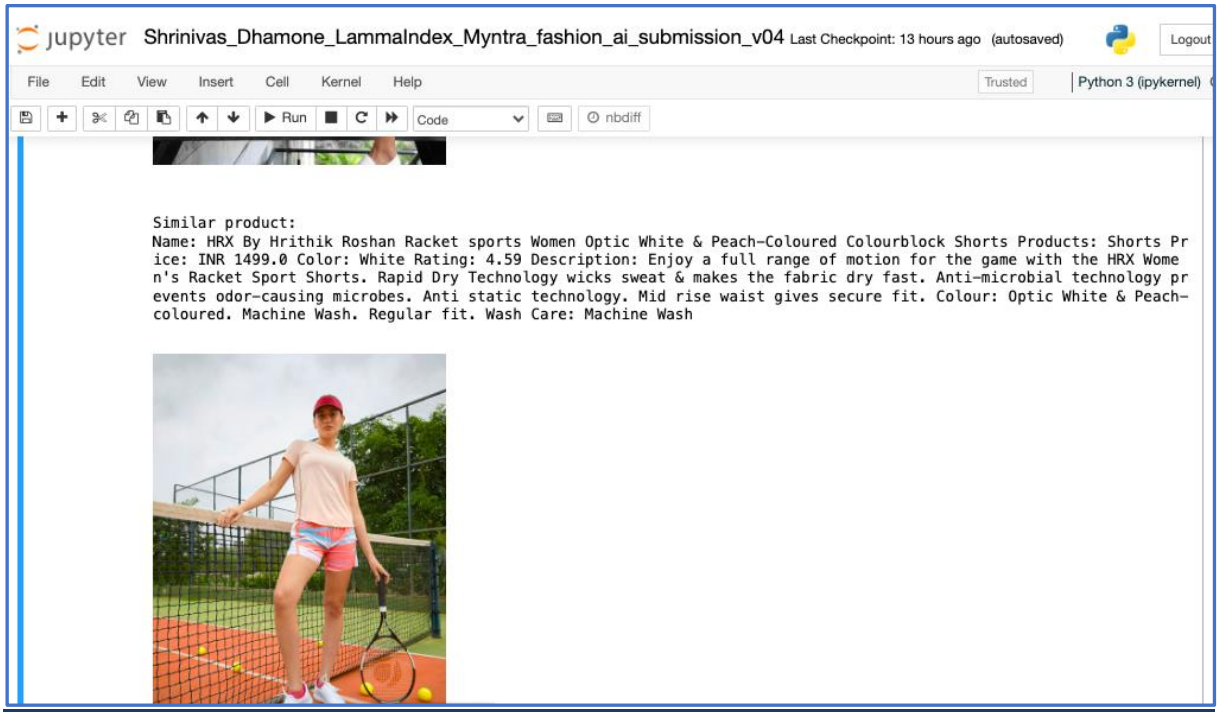
QUERY 1 – SUGGEST ME FLORAL PRINTED DRESS



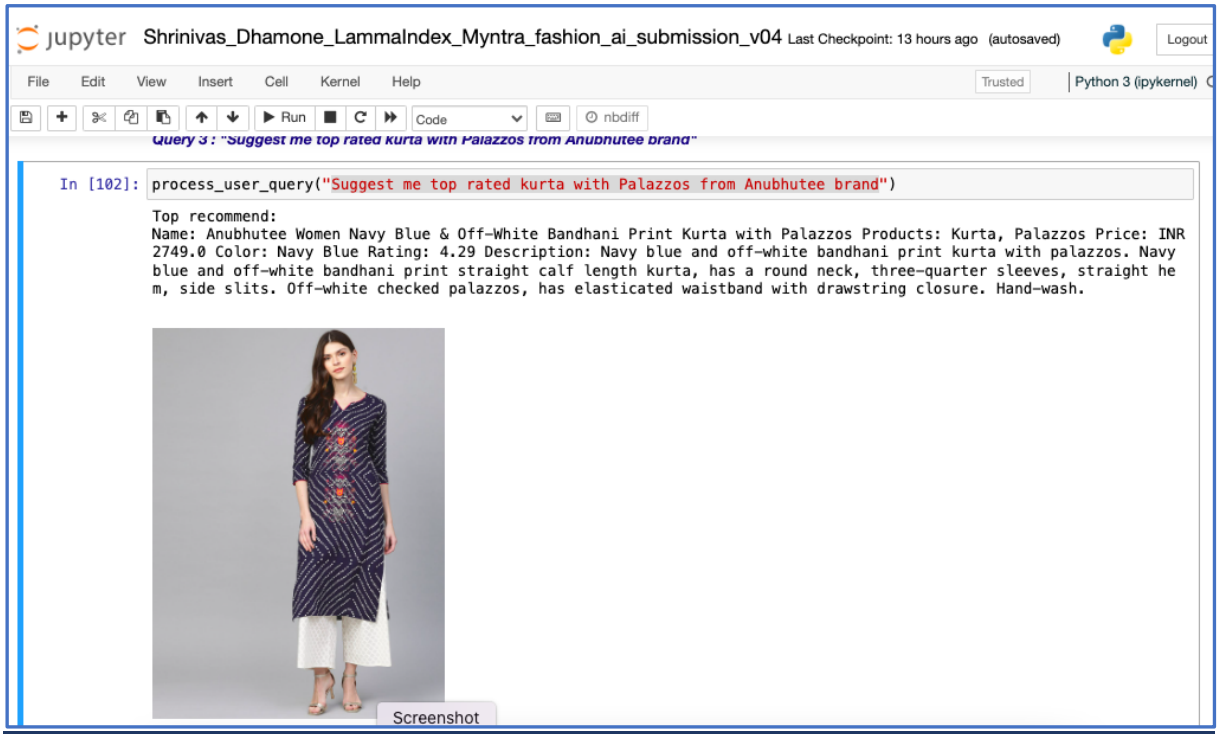


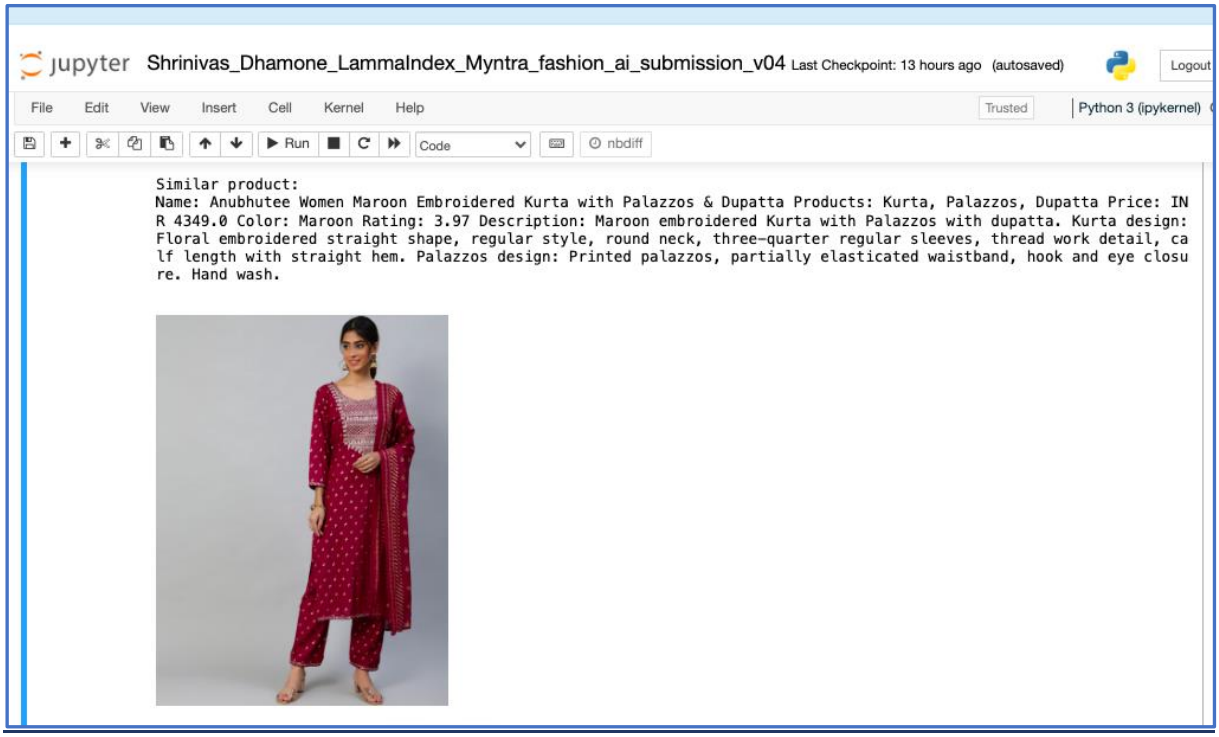
QUERY 2 - SUGGEST ME PEACH COLOURED SHORTS.



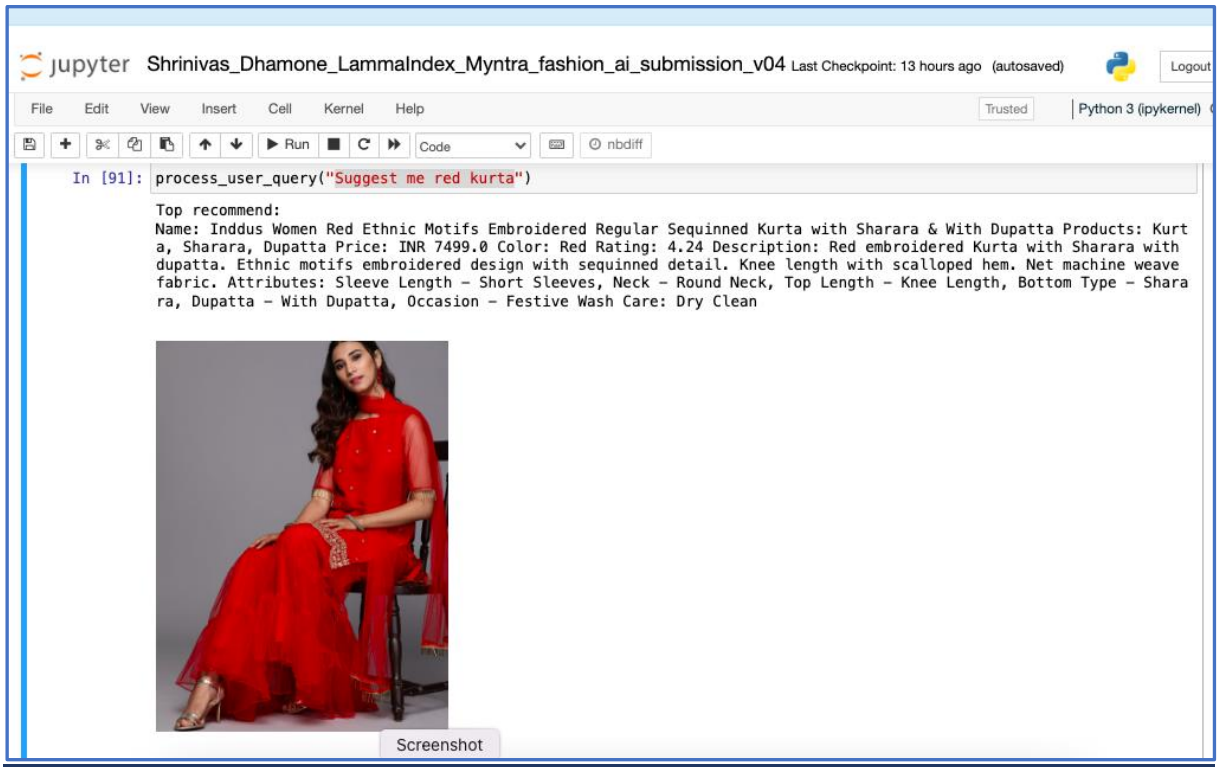


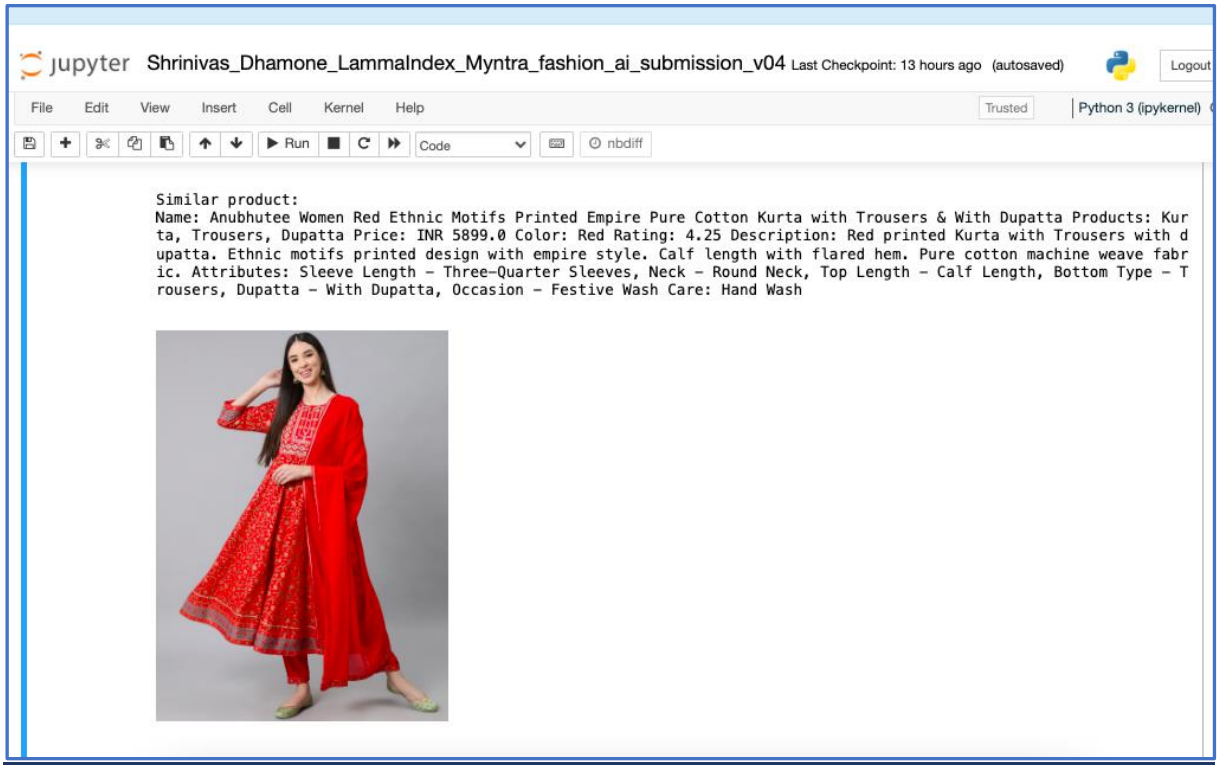
QUERY 3 - SUGGEST ME TOP RATED KURTA WITH PALAZZOS FROM ANUBHUTEE BRAND.



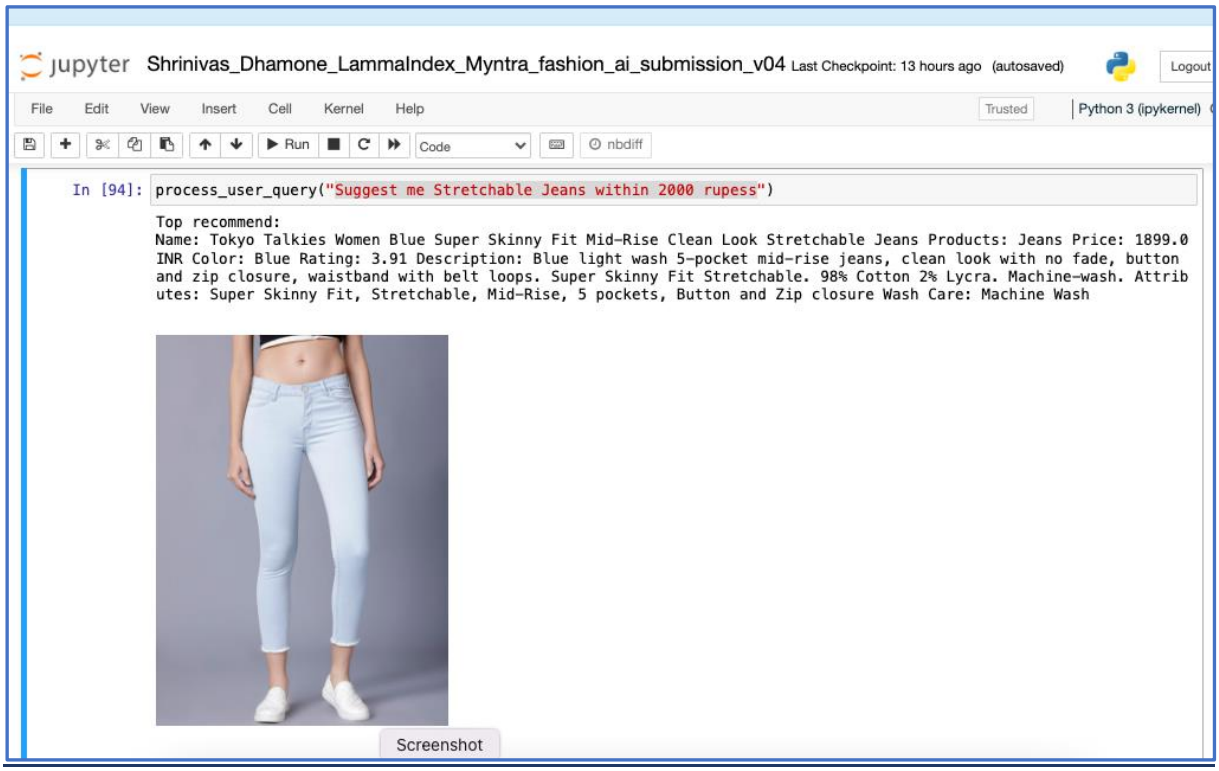


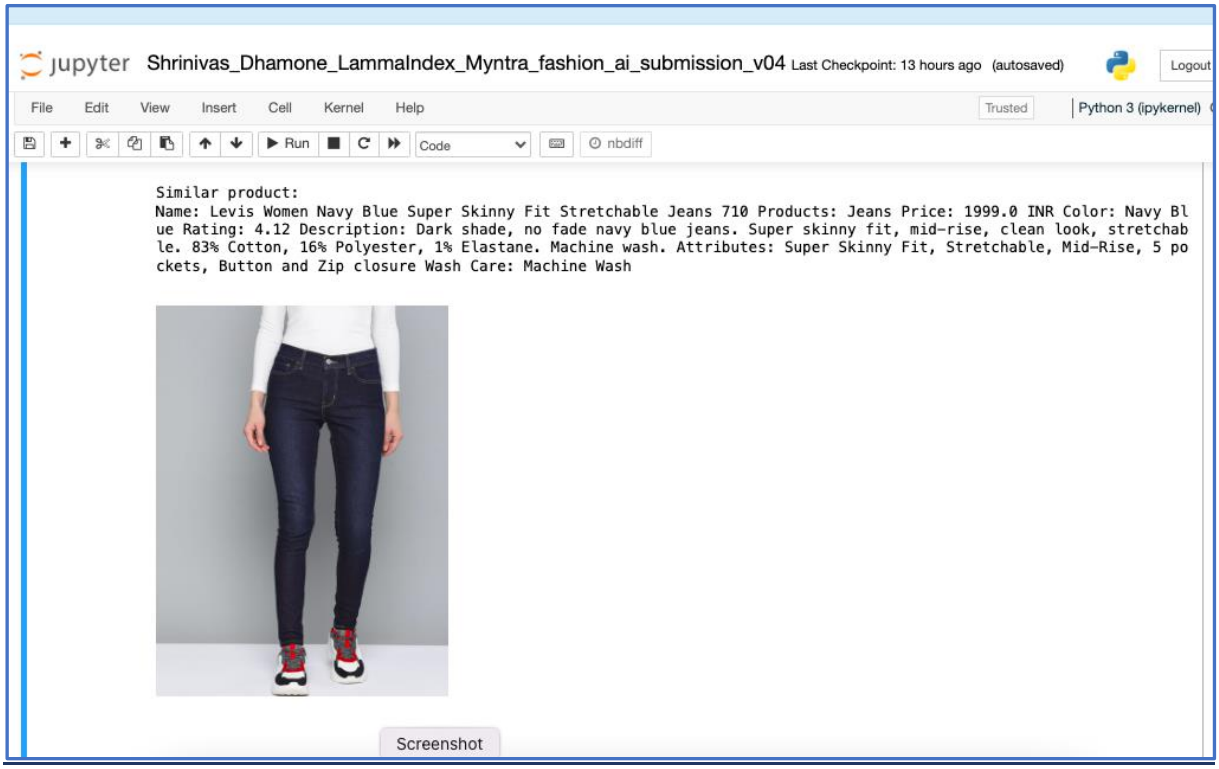
QUERY 4 - SUGGEST ME RED KURTA.





QUERY 5 - SUGGEST ME STRETCHABLE JEANS WITHIN 2000 RUPESS.





QUERY 6 - SUGGEST ME JUMPSUIT IN NAVY BLUE COLOR WITH RATING ABOUE 4

```
Similar product:
Name: Tokyo Talkies Women Navy Blue Solid Basic Jumpsuit Products: Jumpsuit Price: INR 2299.0 Color: Nav
y Blue Rating: 4.31 Description: Navy Blue solid basic jumpsuit with a mandarin collar and three-quarter
sleeves. Cotton material, machine washable. Attributes: Closure - Button, Neck - Mandarin Collar, Sleeve
Length - Three-Quarter Sleeves, Type - Basic Jumpsuit Wash Care: Machine Wash
```

7. SUMMARY

The Fashion Search AI project aims to revolutionize how fashion enthusiasts and industry professionals access and use historical fashion data to make better style choices and predict trends. By employing advanced machine learning techniques and a Retrieval-Augmented Generation (RAG) framework, this system provides an efficient and effective way to quickly search and retrieve relevant information from an extensive fashion archive.

8. EXPLORATION ZONE

Place-holder provided in notebook to explore few more query.

Exploration Zone

- Try out / explore

In [ ]:
```
process_user_query("Pink top above 2500 INR")
```

In [ ]:
```
process_user_query("Floral Top with 4 star rating")
```

In [ ]:
```

```