**Machine Learning Model Deployment with IBM Cloud Watson Studio**

## Product Recommendation System for E-commerce

**OBJECTIVES :**

The primary objective of this project is to develop a personalized product recommendation system for an e-commerce platform. The system will use machine learning algorithms to suggest products to users based on their past interactions, purchase history, and preferences, aiming to enhance user engagement and drive sales

**DEFINITION:**

* Problem: Users struggle with finding relevant products, hindering a seamless shopping experience and reducing sales potential.
* Project Goals: Enhance user engagement and boost conversion rates by delivering personalized product recommendations that align with users' preferences and behaviors.
* Success Metrics: Measure increased click-through rates, prolonged session durations, higher purchase frequency, and improved user satisfaction scores.

**IDEATE:**

* Ideation for the e-commerce recommendation system includes diverse recommendation algorithms (collaborative, content-based, hybrid) and an intuitive user interface featuring personalized homepages, complementary product suggestions, and user profiles. Incorporate feedback mechanisms like thumbs up/down, comments, surveys, and purchase history reviews to optimize recommendations and enhance the user experience.

**EMPATHIZE:**

Understanding user needs and pain points regarding product discovery and shopping experience through surveys, interviews, and user feedback.

**DESIGN THINKING PROCESS:**

* Predictive Use Case: Predicting Customer Product Preferences in E-commerce: Forecast the likelihood of a customer's interest in specific product categories, aiding in personalized product recommendations and targeted marketing strategies.
* Dataset Selection: Utilizing a dataset containing customer profiles, historical purchase data, product attributes, and interactions. This dataset will provide sufficient information to model customer preferences accurately.
* Model Training: Employ a collaborative filtering or hybrid recommendation approach using IBM Cloud Watson Studio. Utilize the dataset to train the model, capturing user-product interactions and preferences.
* Model Deployment: Deploy the trained recommendation model using IBM Cloud Watson Studio's deployment capabilities, converting it into a web service accessible via an API.
* Integration: Integrate the deployed model into the e-commerce platform's backend.

**Project Objectives:**

**Objective:** Develop a Personalized Product Recommendation System for an E-commerce Platform.

**Objective Details:** The primary goal of this project is to build a personalized product recommendation system that utilizes machine learning algorithms. This system will offer product suggestions to users based on their past interactions, purchase history, and preferences, ultimately enhancing user engagement and driving sales.

**Problem:** Users currently encounter difficulties in finding relevant products, leading to a suboptimal shopping experience and potential sales loss.

**Project Goals:** The project aims to enhance user engagement and boost conversion rates by delivering personalized product recommendations that align with users' preferences and behaviours.

**Success Metrics:** The success of the project will be measured through increased click-through rates, prolonged session durations, higher purchase frequency, and improved user satisfaction scores.

**Ideation and Design Thinking Process**

**Ideation:** The ideation phase for the E-commerce Recommendation System involves exploring various recommendation algorithms, including collaborative, content-based, and hybrid models. Additionally, it entails designing an intuitive user interface that incorporates personalized homepages, complementary product suggestions, and user profiles. The system will also feature feedback mechanisms such as thumbs up/down, comments, surveys, and purchase history reviews to continually optimize recommendations and enhance the user experience.

**Understanding User Needs:** To inform the system's design, it's crucial to empathize with users by understanding their needs and pain points related to product discovery and the shopping experience. This understanding will be gathered through surveys, interviews, and user feedback, enabling the system to cater to user preferences effectively.

**Predictive Use Case**

**Use Case:** Predicting Customer Product Preferences in E-commerce.

**Use** **Case Details:** This use case involves forecasting the likelihood of a customer's interest in specific product categories. This prediction is instrumental in providing personalized product recommendations and enabling targeted marketing strategies.

**Dataset Selection:** The project will utilize a comprehensive dataset containing customer profiles, historical purchase data, product attributes, and user interactions. This dataset will provide the necessary information to accurately model customer preferences.

**Model Training:** The recommendation system will employ either a collaborative filtering or hybrid recommendation approach using IBM Cloud Watson Studio. The dataset will be used to train the model, capturing user-product interactions and preferences. Model Deployment: Once trained, the recommendation model will be deployed using IBM Cloud Watson Studio's deployment capabilities, transforming it into a web service accessible via an API.

**Integration:** The deployed model will be seamlessly integrated into the e-commerce platform's backend, ensuring a smooth user experience and facilitating real-time product recommendations based on user behaviour and preferences

The recommendation system I've designed is tailored to the customer journey of individuals who visit an e-commerce website, aiming to enhance their experience and boost customer acquisition and retention. This system is structured into three components to align with the specific needs of the business:

**Part I:** Popularity-Based Recommendation System for New Customers

When a new customer arrives on the website without any prior purchase history, they are initially presented with product recommendations based on the overall popularity of items. These recommendations help newcomers discover the most sought-after products available on the website.

**Part II:** Model-Based Collaborative Filtering System

Once the new customer makes their first purchase , the recommendation system transitions to a more personalized approach. It leverages collaborative filtering techniques, which take into account the customer's purchase history and incorporate ratings provided by other users who have bought similar items. This results in tailored product recommendations based on the customer's individual preferences and behaviour.

**Part III:** Recommendations for Businesses without Product Ratings

In scenarios where a business is setting up its e-commerce website and lacks product ratings, the recommendation system is capable of offering valuable suggestions based on alternative data points and customer behaviour.

By dividing the recommendation system into these three parts, we ensure that it caters to the diverse needs of customers, whether they are newcomers or returning shoppers, and whether or not product ratings are available. This approach ultimately contributes to an improved customer experience and supports the business in acquiring and retaining customers effectively.

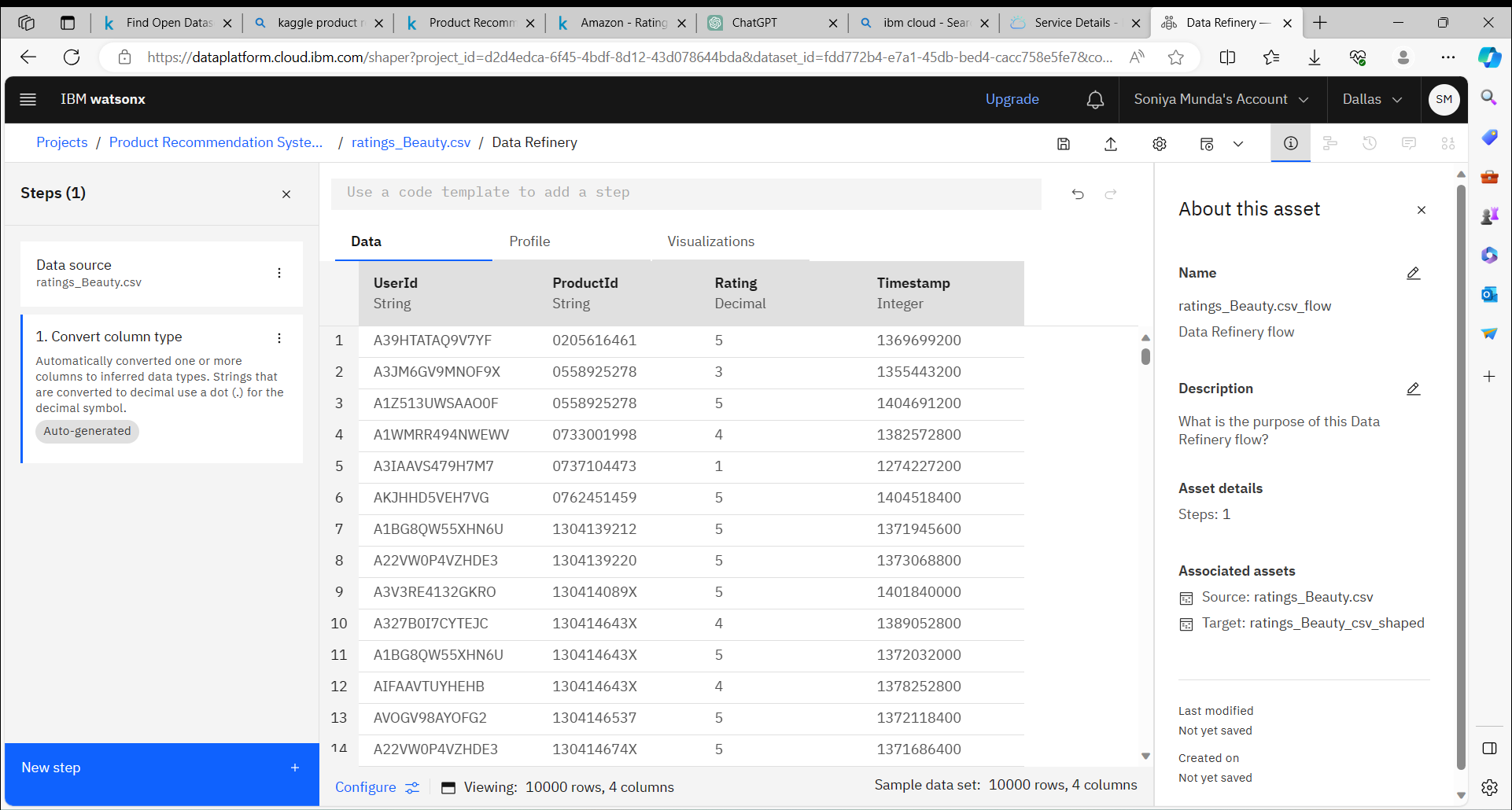
## Part I: Popularity-Based Recommendation System for New Customers

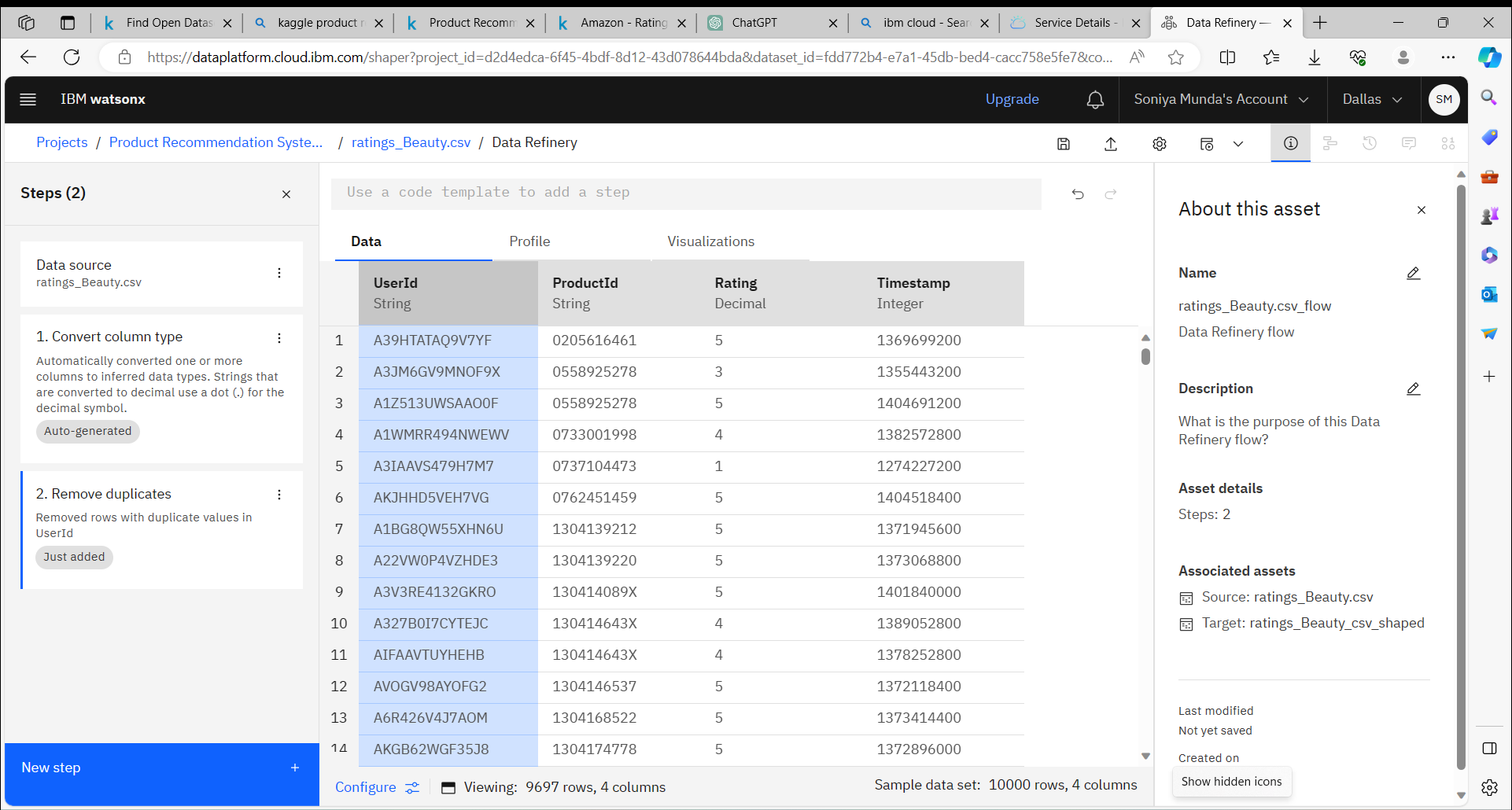
Utilizing a popularity-based strategy to engage new customers with the top-selling products on an e-commerce website is a highly effective approach, especially when launching a recommendation engine from scratch or dealing with limited data.

**Dataset:** Amazon Product Review Dataset

To implement this recommendation system, we begin by working with the Amazon product review dataset, a valuable source of information for product popularity and customer preferences.

**Data cleaning:**



**Removing Duplicates:**

#### Importing libraries:

The first step involves importing the necessary libraries and tools to build and execute the recommendation system. This step ensures that we have the required resources at our disposal to work with the dataset and create meaningful recommendations.

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

# %matplotlib inline

plt.style.use("ggplot")

import sklearn

from sklearn.decomposition import TruncatedSVD

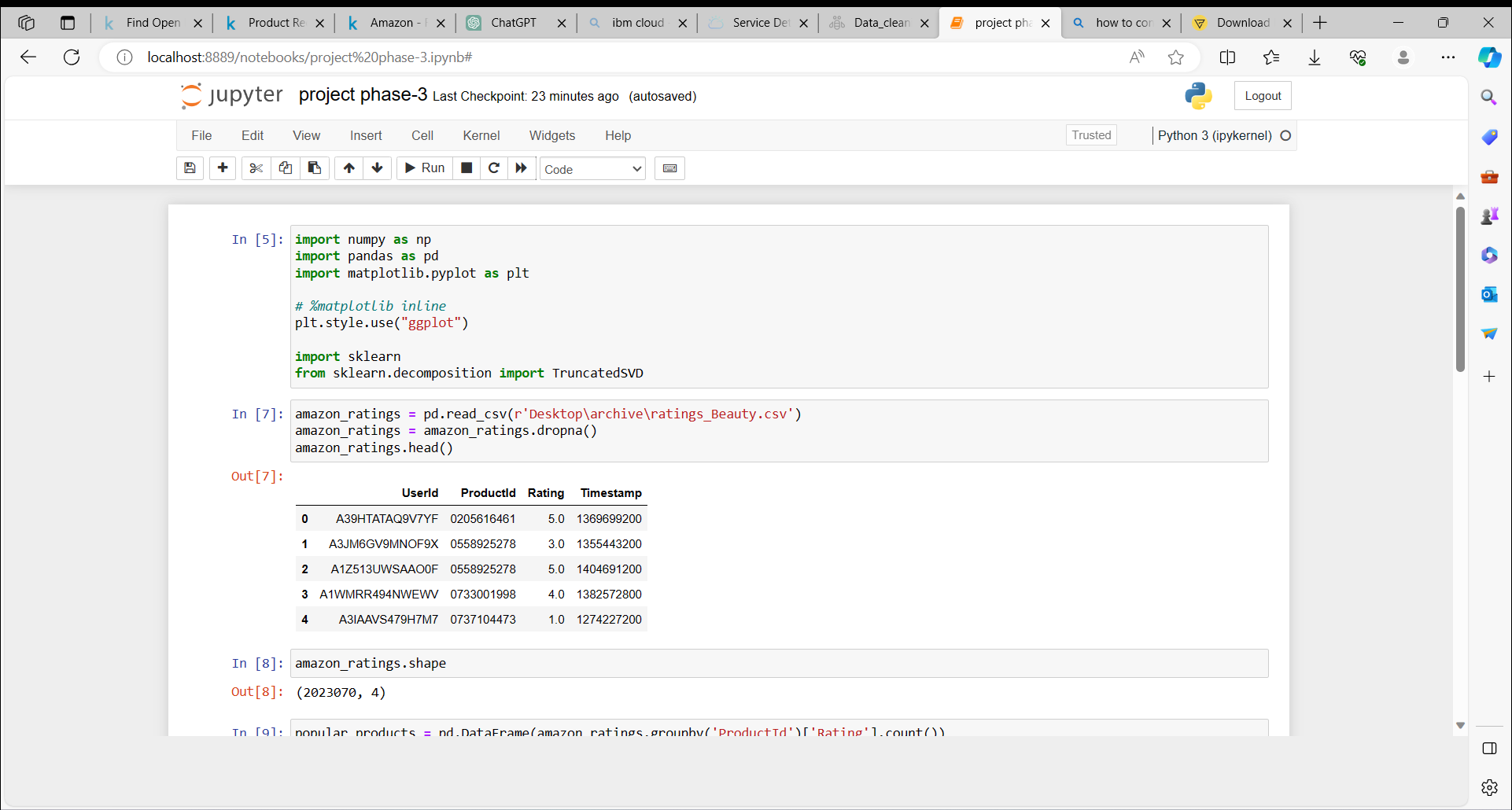
#### Loading the dataset:

amazon\_ratings = pd.read\_csv('r Desktop\archive\ratings\_Beauty.csv')

amazon\_ratings = amazon\_ratings.dropna()

amazon\_ratings.head()

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | UserId | ProductId | Rating | Timestamp |
| 0 | A39HTATAQ9V7YF | 0205616461 | 5.0 | 1369699200 |
| 1 | A3JM6GV9MNOF9X | 0558925278 | 3.0 | 1355443200 |
| 2 | A1Z513UWSAAO0F | 0558925278 | 5.0 | 1404691200 |
| 3 | A1WMRR494NWEWV | 0733001998 | 4.0 | 1382572800 |
| 4 | A3IAAVS479H7M7 | 0737104473 | 1.0 | 1382572800 |



amazon\_ratings.shape

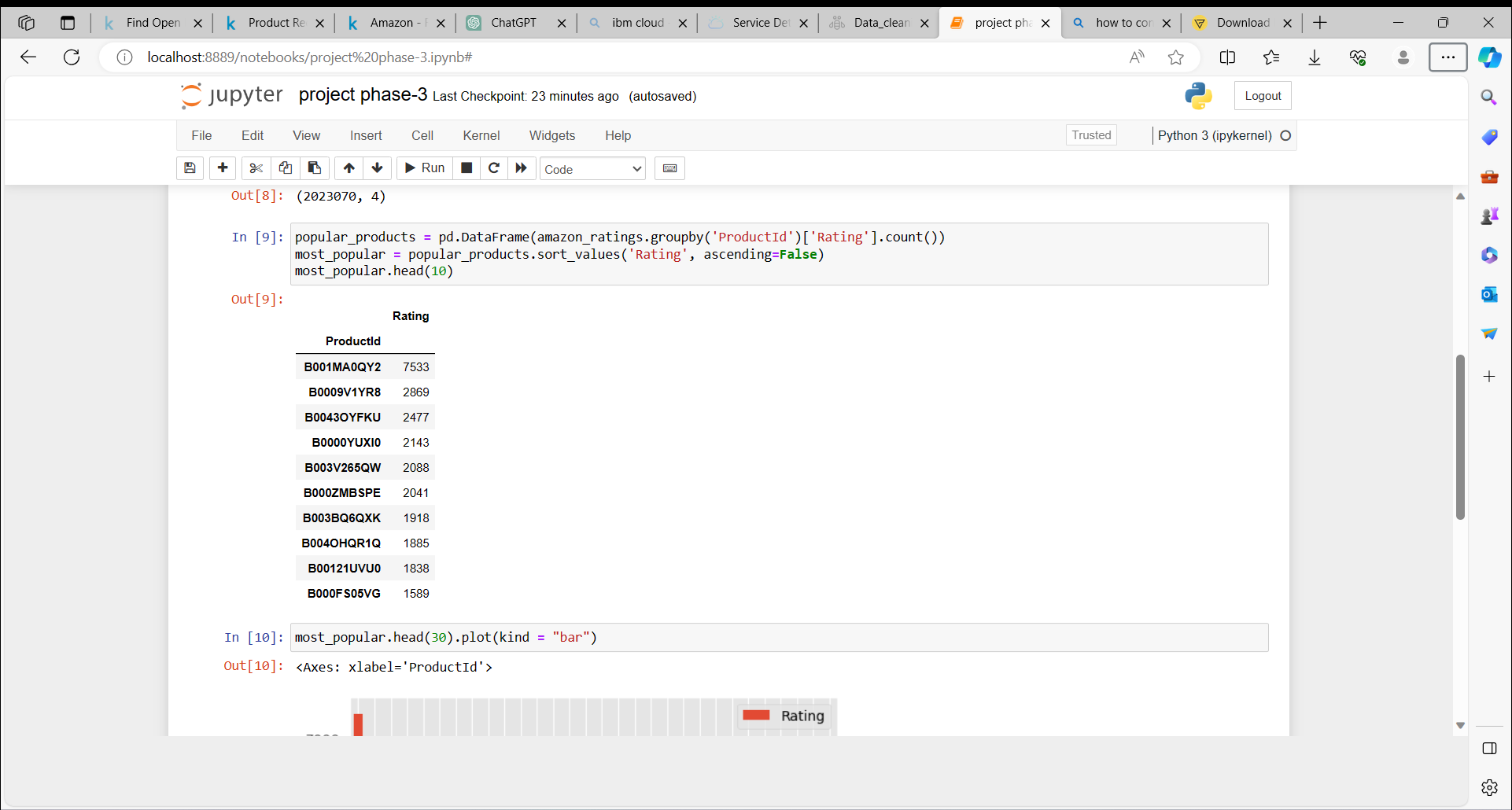
(2023070, 4)

popular\_products = pd.DataFrame(amazon\_ratings.groupby('ProductId')['Rating'].count())

most\_popular = popular\_products.sort\_values('Rating', ascending=False)

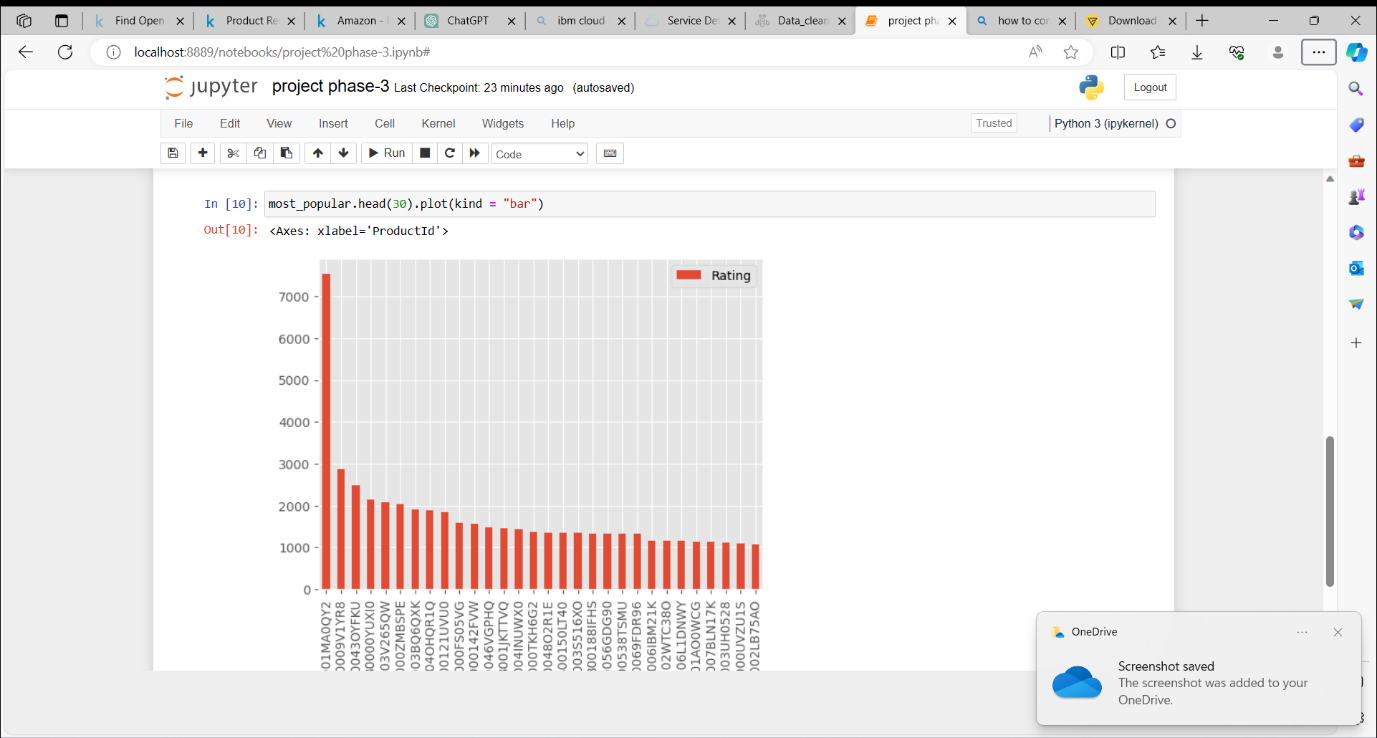
most\_popular.head(10)

| ProductId | Rating |
| --- | --- |
| B001MA0QY2 | 7533 |
| B0009V1YR8 | 2869 |
| B0043OYFKU | 2477 |
| B0000YUXI0 | 2143 |
| B003V265QW | 2088 |
| B000ZMBSPE | 2041 |
| B003BQ6QXK | 1918 |
| B004OHQR1Q | 1885 |
| B00121UVU0 | 1838 |
| B000FS05VG | 1589 |



most\_popular.head(30).plot(kind = "bar")

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fd439e493c8>



**Analysis:**

* The above graph gives us the most popular products (arranged in descending order) sold by the business.
* For example, product, ID # B001MA0QY2 has sales of over 7000, the next most popular product, ID # B0009V1YR8 has sales of 3000, etc.

# Part II: Model-Based Collaborative Filtering System

A model-based collaborative filtering system is used to recommend items to users by analyzing their purchase history and comparing the ratings given by other users who have bought similar items. This approach leverages patterns in user preferences from a diverse set of user data.

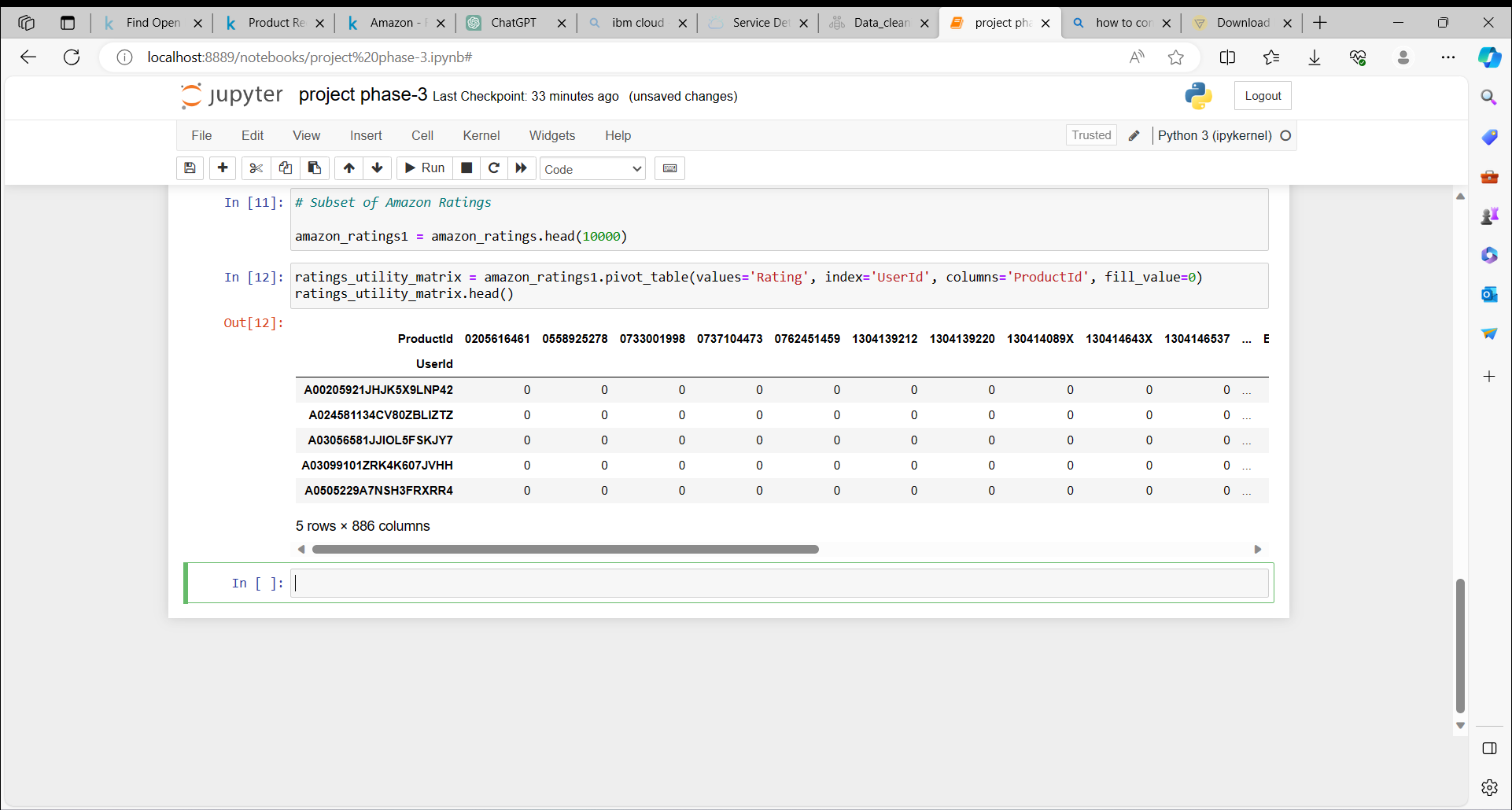
The utility matrix, which is the foundation of this system, is a representation of user-item preferences in the form of a matrix. This matrix contains information about the ratings and preferences of users for various items. Since users typically do not purchase or rate all the items available, the utility matrix is sparse, with many values remaining unknown.

# Subset of Amazon Ratings

amazon\_ratings1 = amazon\_ratings.head(10000)

ratings\_utility\_matrix = amazon\_ratings1.pivot\_table(values='Rating', index='UserId', columns='ProductId', fill\_value=0)

ratings\_utility\_matrix.head()



As expected, the utility matrix obtaned above is sparce, I have filled up the unknown values wth 0.

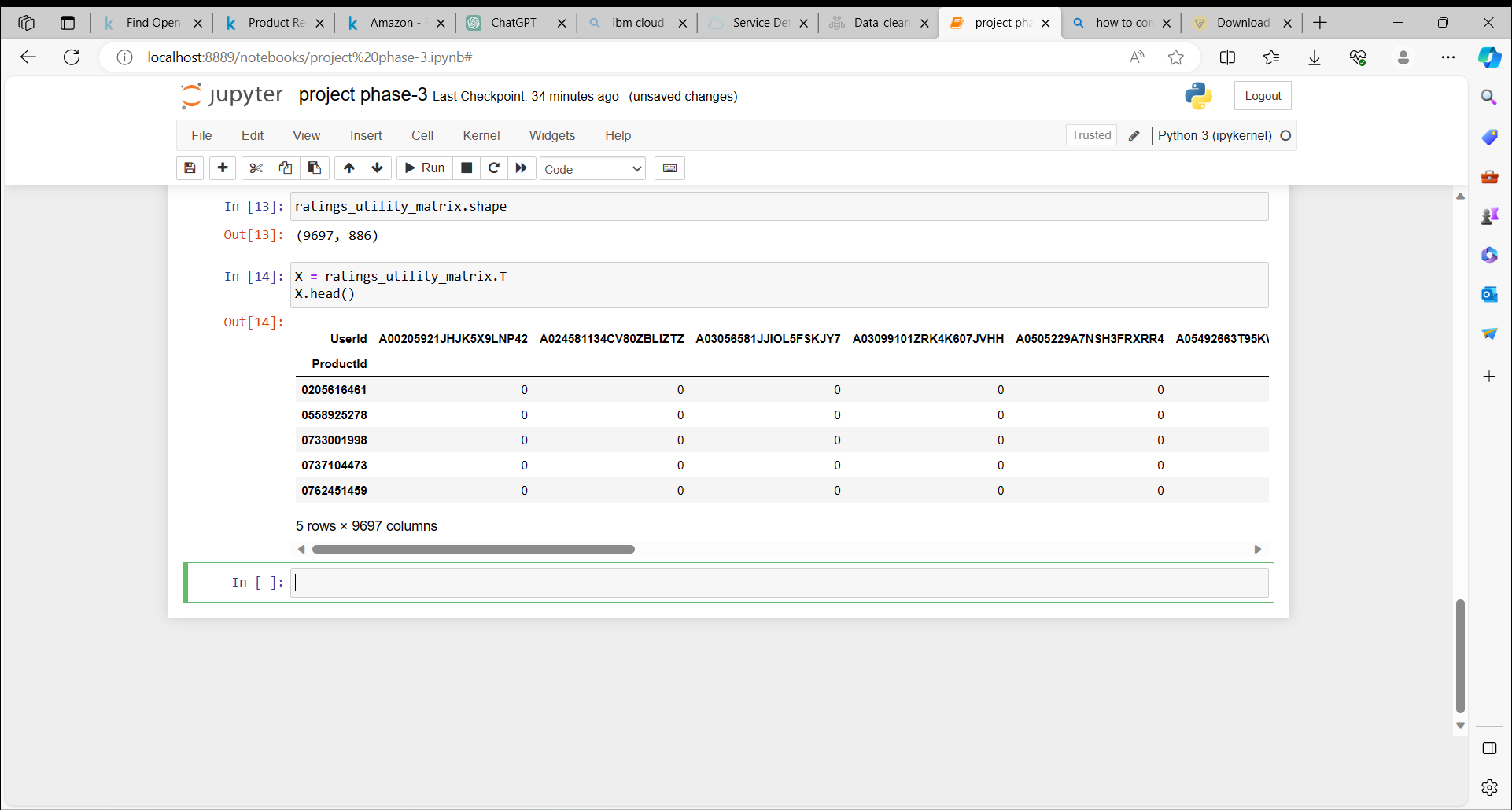
ratings\_utility\_matrix.shape

(9697, 886)

Transposing the matrix

X = ratings\_utility\_matrix.T

X.head()



X.shape

(886, 9697)

Unique products in subset of data

X1 = X

### Decomposing the Matrix

SVD = TruncatedSVD(n\_components=10)

decomposed\_matrix = SVD.fit\_transform(X)

decomposed\_matrix.shape

(886, 10)

### Correlation Matrix

correlation\_matrix = np.corrcoef(decomposed\_matrix)

correlation\_matrix.shape

(886, 886)

### Isolating Product ID # 6117036094 from the Correlation Matrix

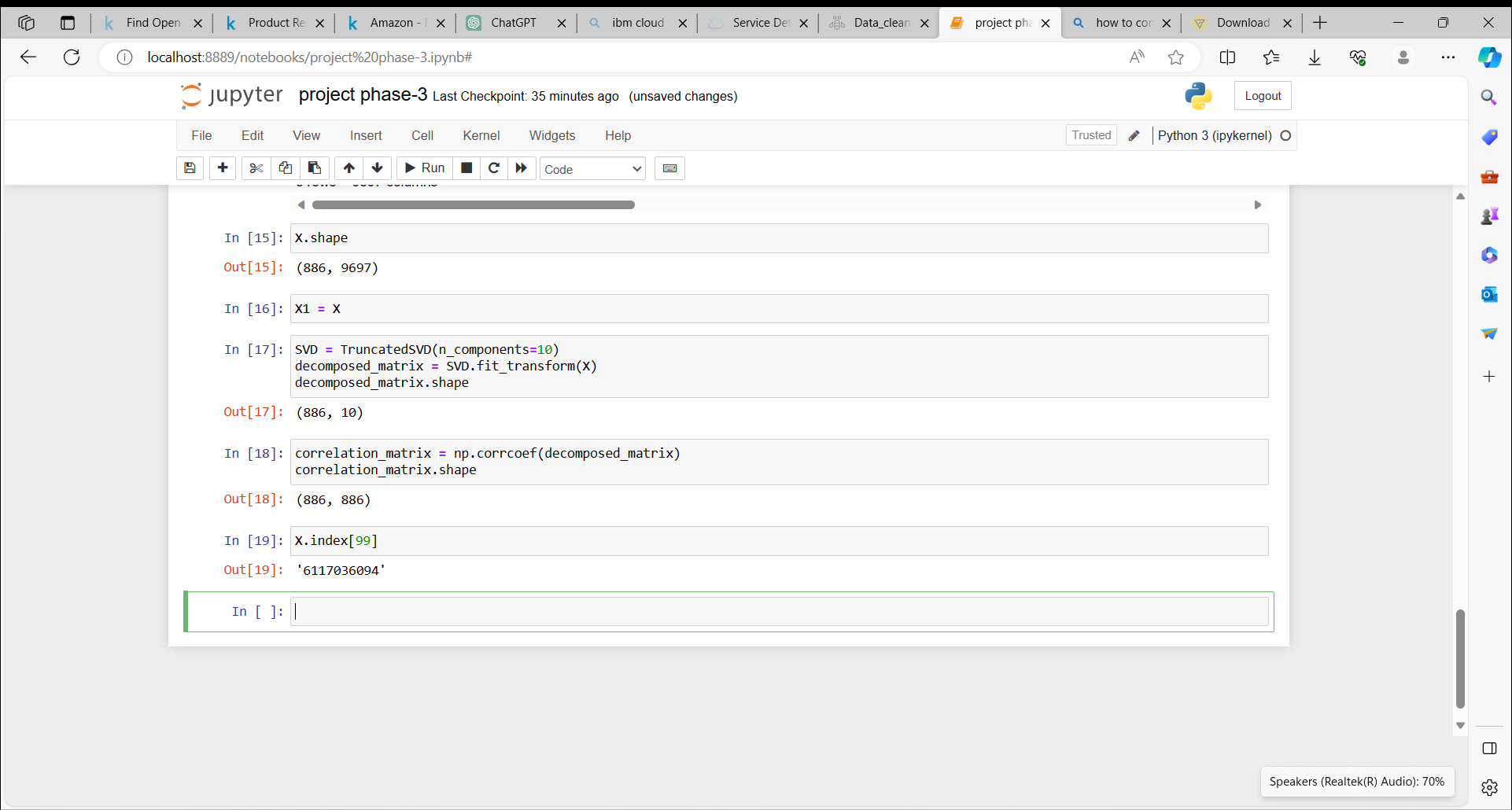
Assuming the customer buys Product ID # 6117036094 (randomly chosen)

X.index[99]

'6117036094'

linkcode

Index # of product ID purchased by customer



i = "6117036094"

product\_names = list(X.index)

product\_ID = product\_names.index(i)

product\_ID

99

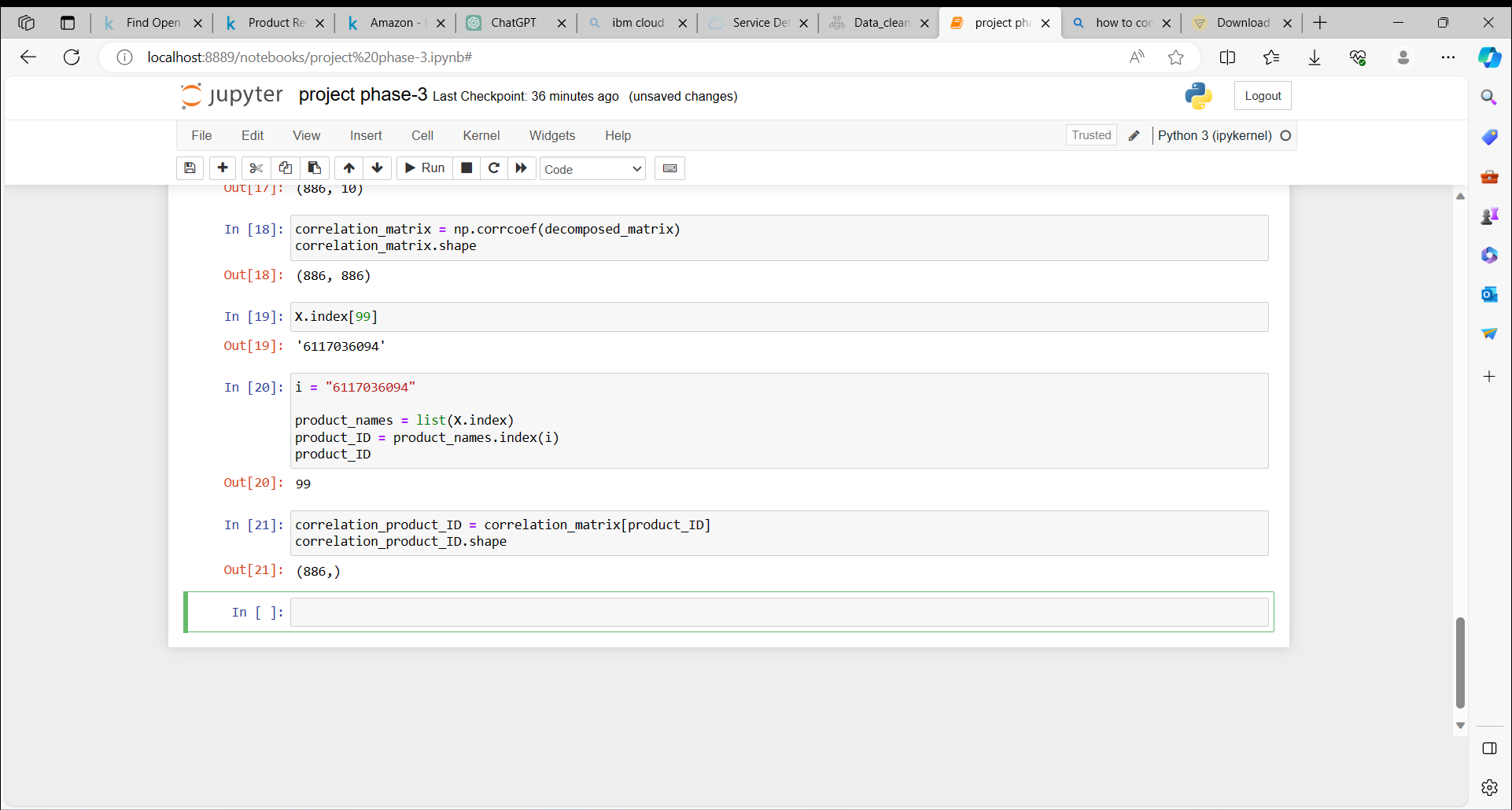
Linkcode

Correlation for all items with the item purchased by this customer based on items rated by other customers people who bought the same product

correlation\_product\_ID = correlation\_matrix[product\_ID]

correlation\_product\_ID.shape

(886,)



### Recommending top 10 highly correlated products in sequence

Recommend = list(X.index[correlation\_product\_ID > 0.90])

*# Removes the item already bought by the customer*

Recommend.remove(i)

Recommend[0:9]

['0733001998',

'1304139212',

'1304139220',

'130414089X',

'130414643X',

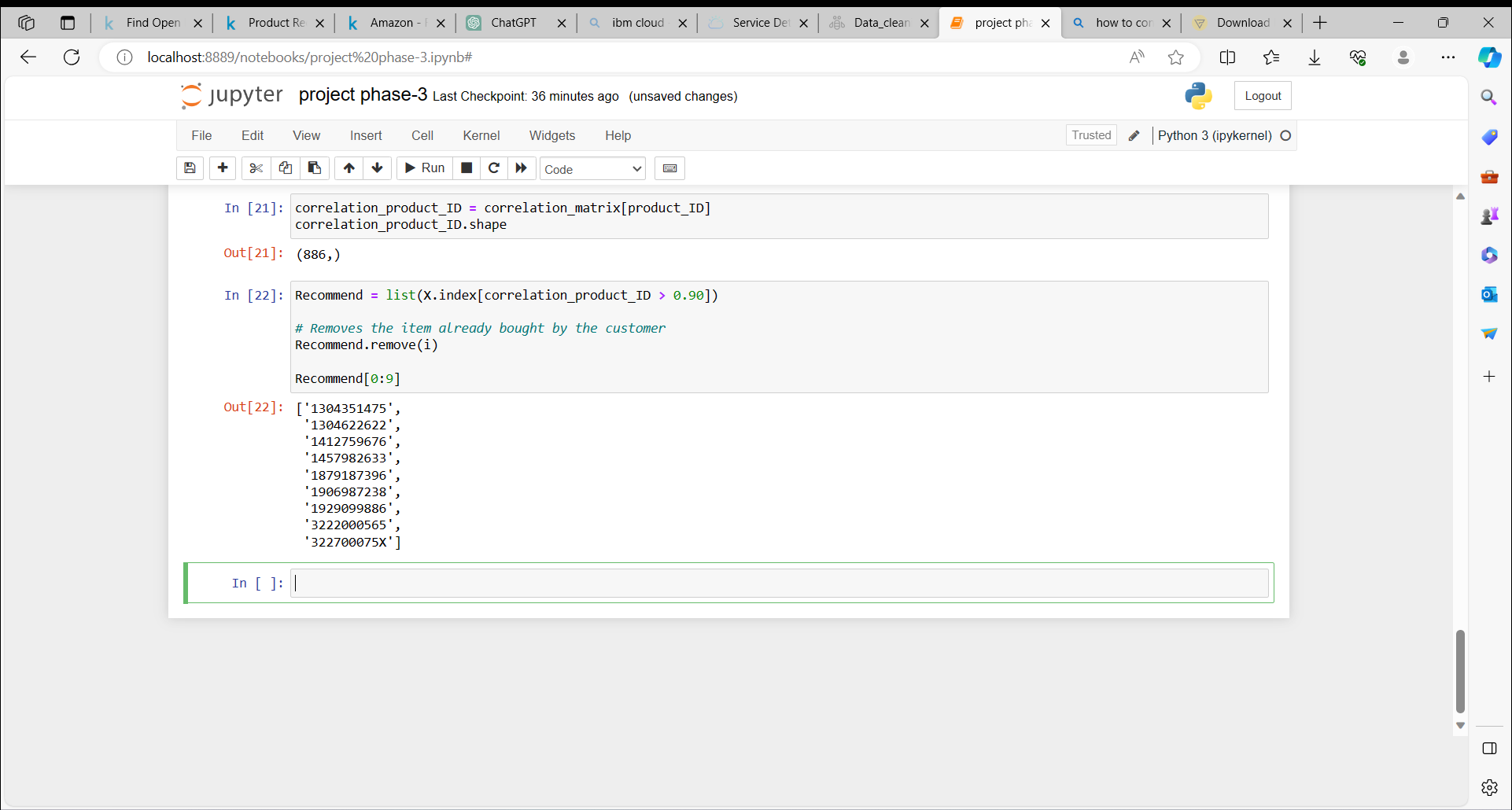
'130414674X',

'1304174778',

'1304174867',

'1304174905']

**Product Id #**Here are the top 10 products to be displayed by the recommendation system to the above customer based on the purchase history of other customers in the website.



# Part III: Recommendations for Businesses Without Product Ratings

In the absence of user-item purchase history, a recommendation system for a business can be created using a search engine approach. This system generates product recommendations by analyzing and clustering textual information provided in product descriptions.

To accomplish this, a dataset from Home Depot, containing product information, is utilized as the source of data. This dataset serves as the foundation for generating recommendations based on the textual content and characteristics of the products.

# Importing libraries

from sklearn.feature\_extraction.text import TfidfVectorizer, CountVectorizer

from sklearn.neighbors import NearestNeighbors

from sklearn.cluster import KMeans

from sklearn.metrics import adjusted\_rand\_score

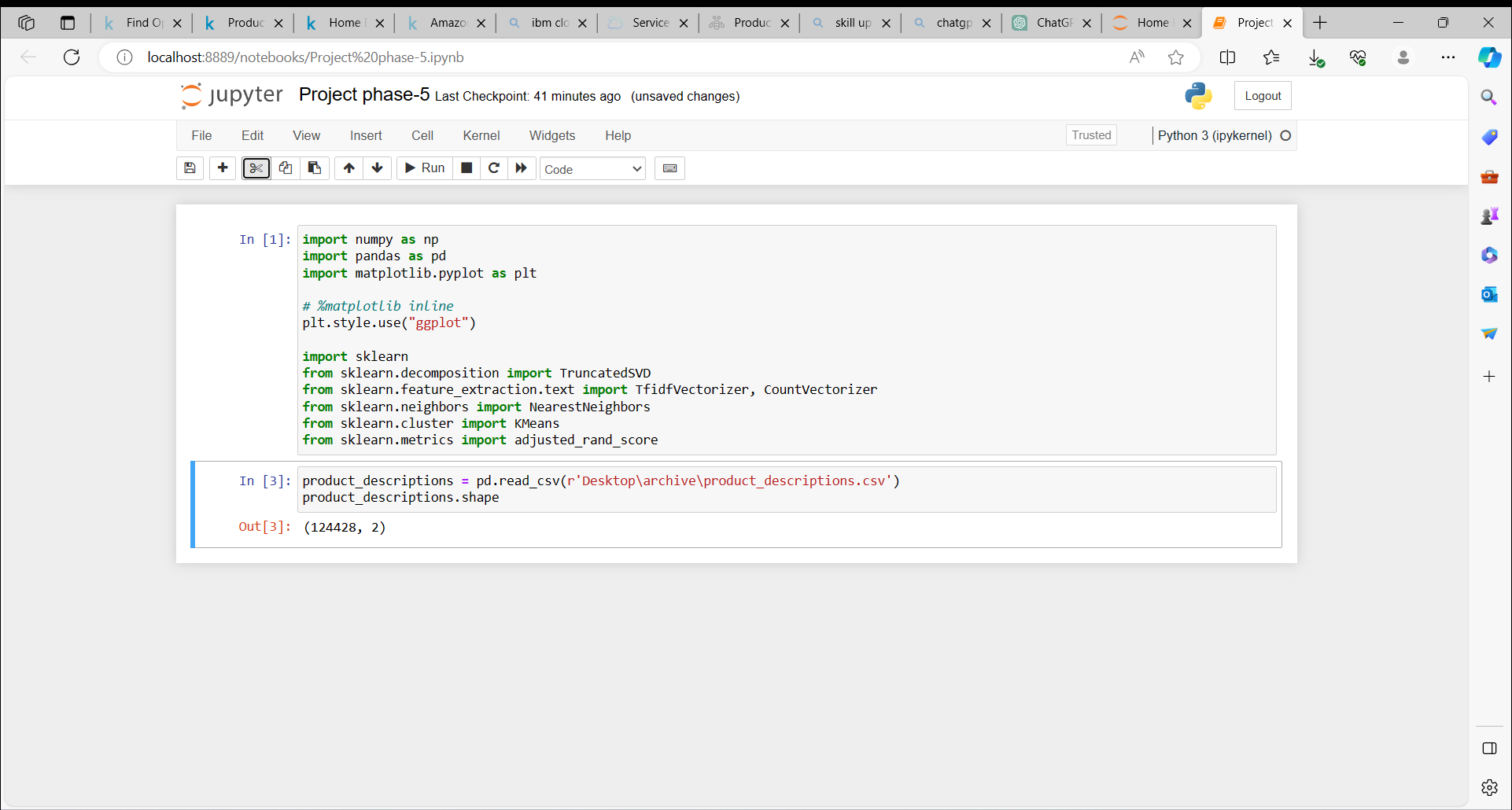
### Item to item based recommendation system based on product description

Applicable when business is setting up its E-commerce website for the first time

product\_descriptions = pd.read\_csv(r’Desktop\archive\product\_descriptions.csv')

product\_descriptions.shape

(124428, 2)



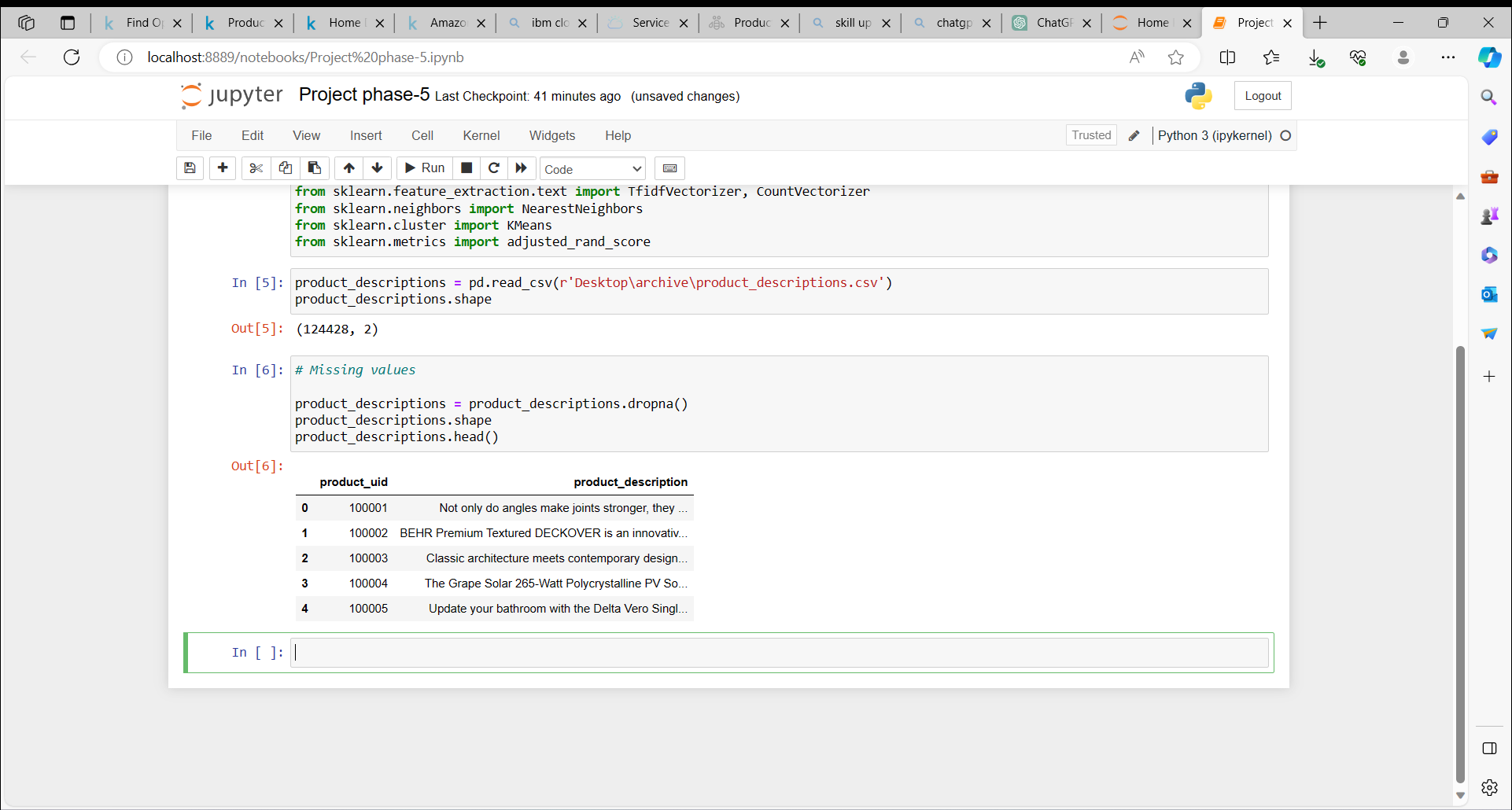
#### Checking for missing values

# Missing values

product\_descriptions = product\_descriptions.dropna()

product\_descriptions.shape

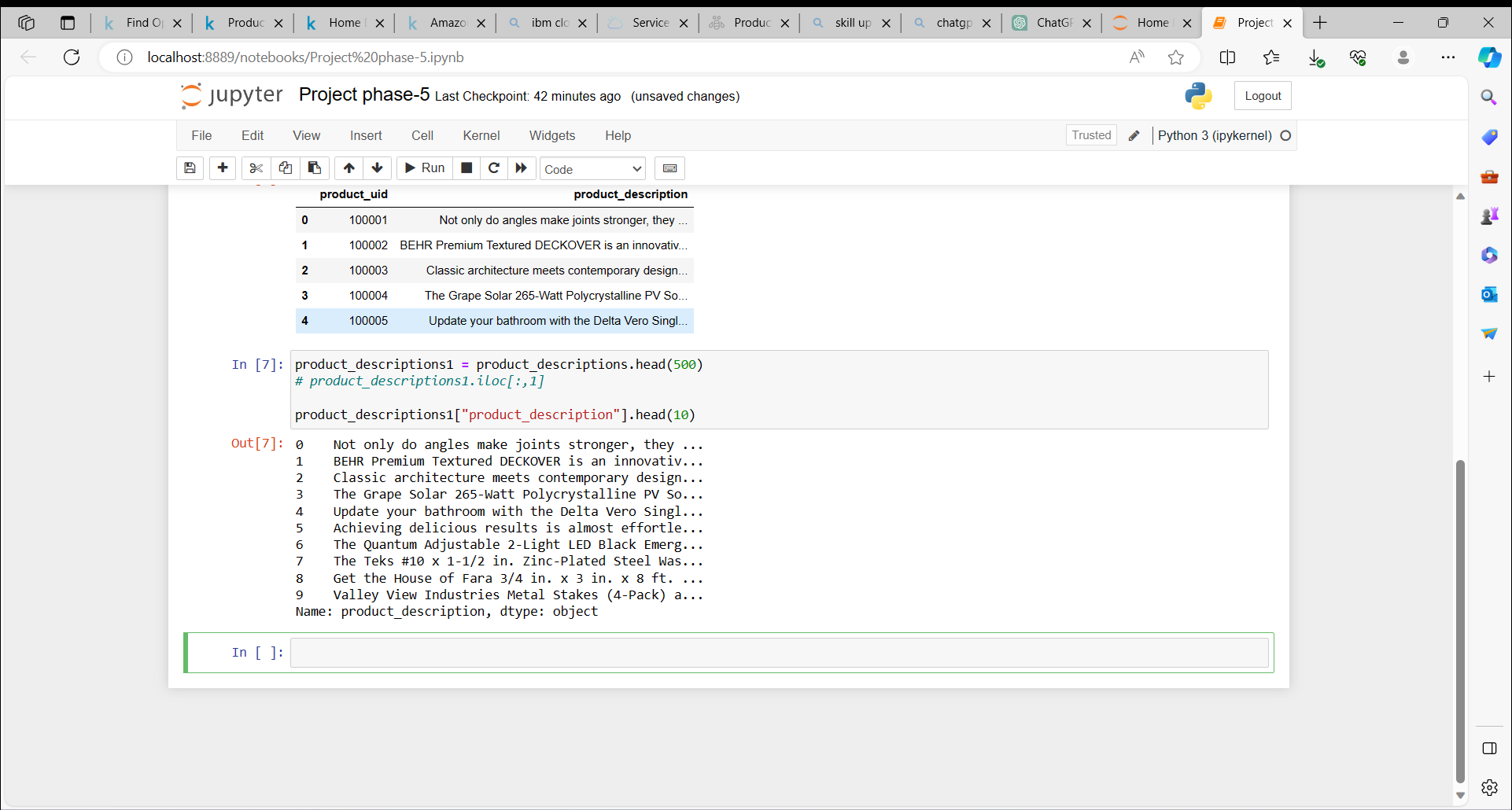
product\_descriptions.head()



product\_descriptions1 = product\_descriptions.head(500)

# product\_descriptions1.iloc[:,1]

product\_descriptions1["product\_description"].head(10)



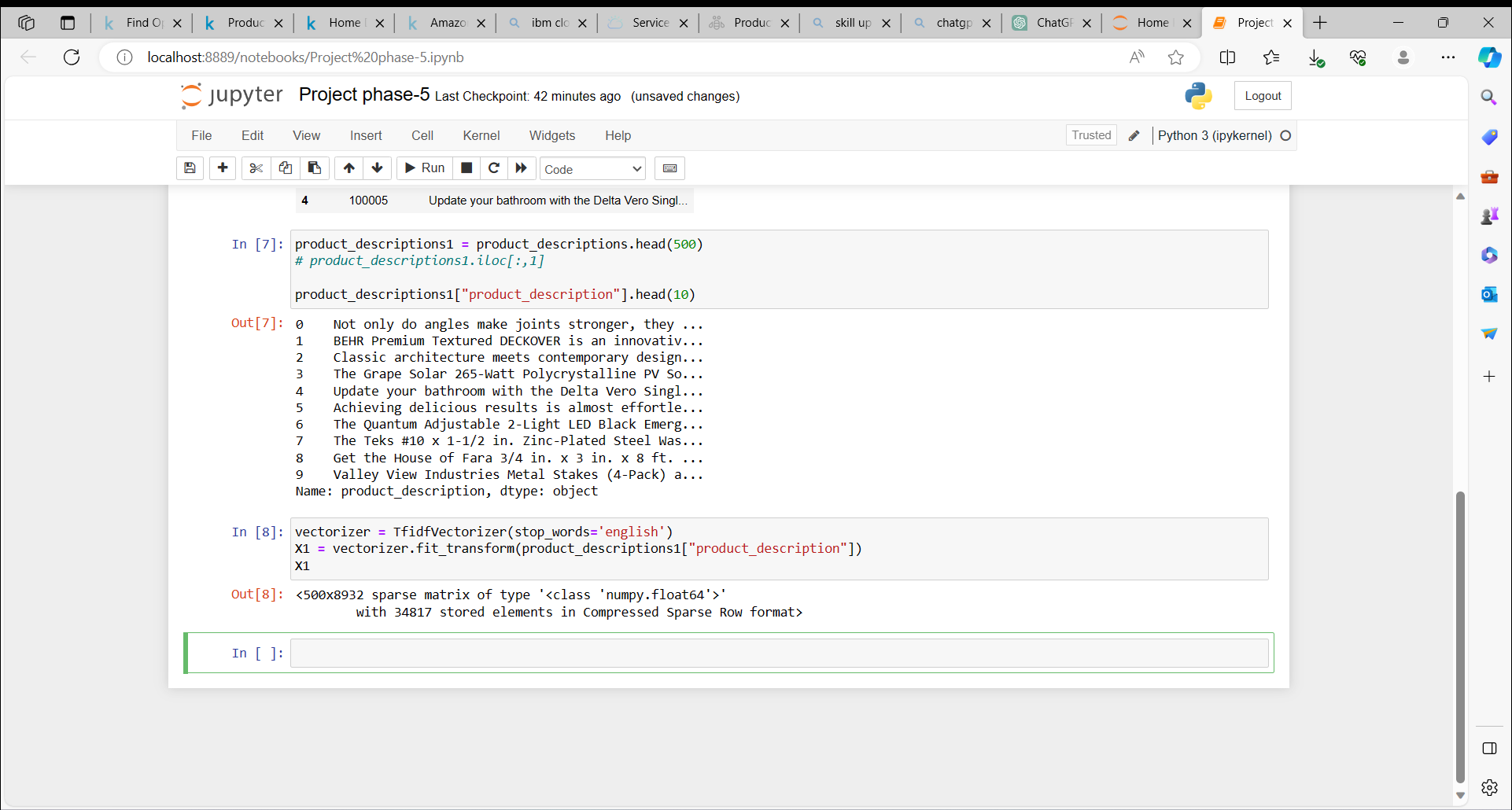
#### Feature extraction from product descriptions

Converting the text in product description into numerical data for analysis

vectorizer = TfidfVectorizer(stop\_words='english')

X1 = vectorizer.fit\_transform(product\_descriptions1["product\_description"])

X1



#### Visualizing product clusters in subset of data

**linkcode**

**# Fitting K-Means to the dataset**

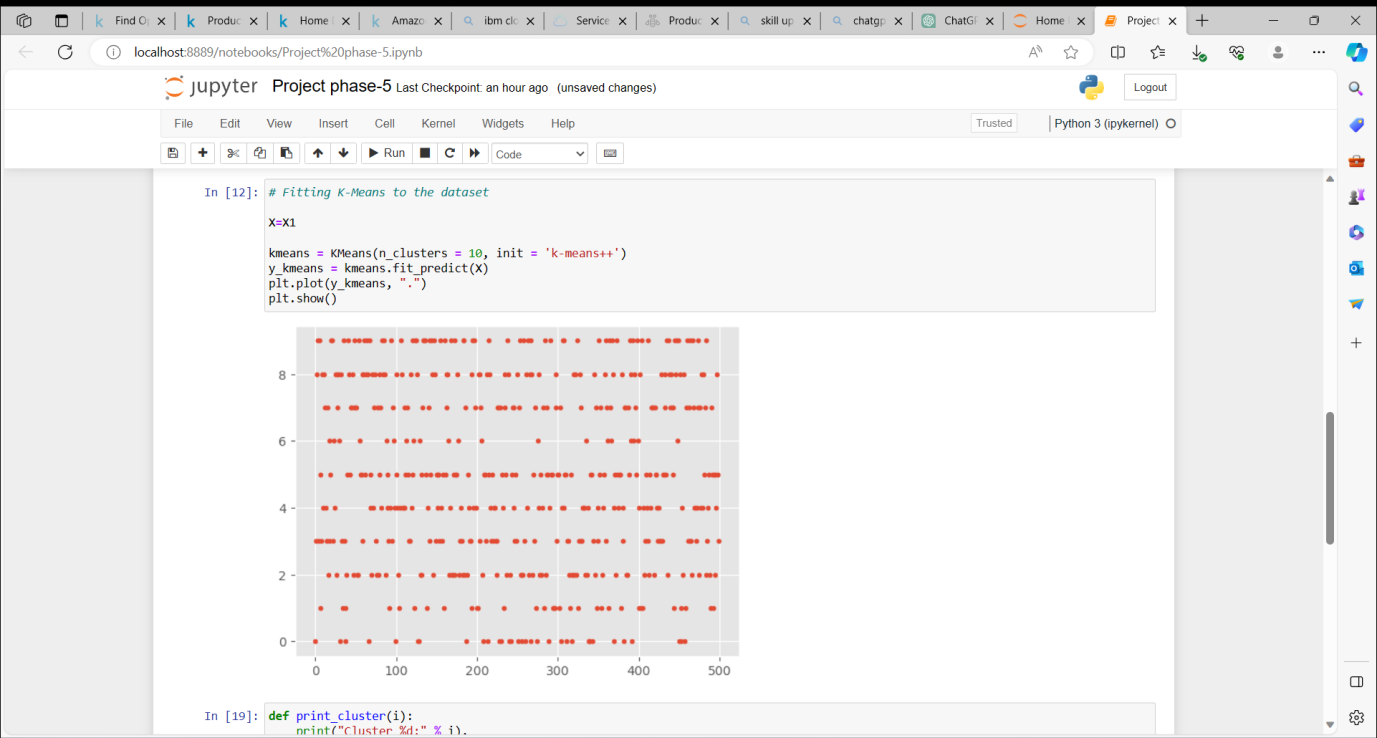
X=X1

kmeans = KMeans(n\_clusters = 10, init = 'k-means++')

y\_kmeans = kmeans.fit\_predict(X)

plt.plot(y\_kmeans, ".")

plt.show()



def print\_cluster(i):

print("Cluster %d:" % i),

for ind in order\_centroids[i, :10]:

print(' %s' % terms[ind]),

print

* Recommendation of product based on the current product selected by user.
* To recommend related product based on, Frequently bought together.

#### Top words in each cluster based on product description

# # Optimal clusters is

true\_k = 10

model = KMeans(n\_clusters=true\_k, init='k-means++', max\_iter=100, n\_init=1)

model.fit(X1)

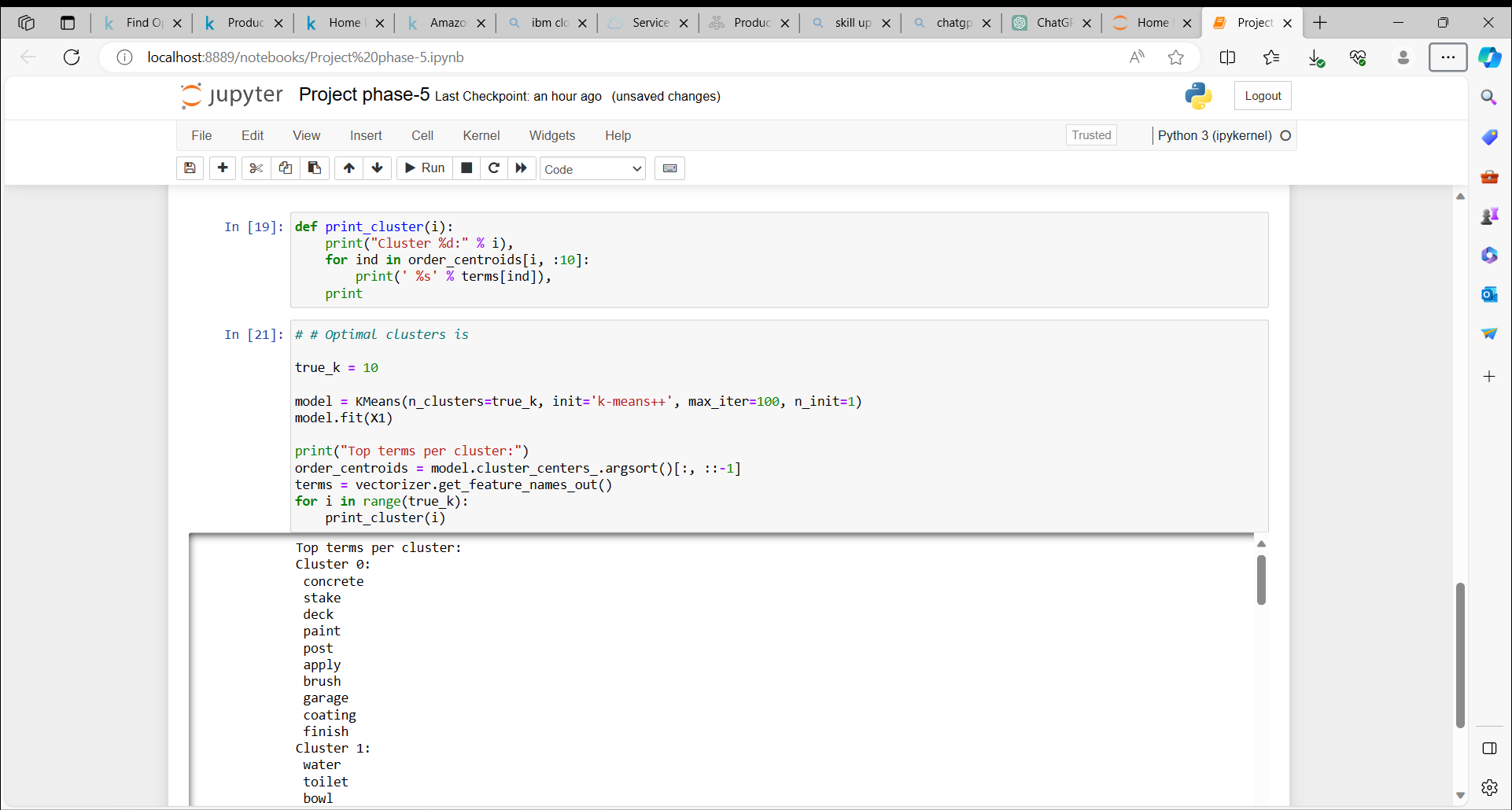
print("Top terms per cluster:")

order\_centroids = model.cluster\_centers\_.argsort()[:, ::-1]

terms = vectorizer.get\_feature\_names()

for i in range(true\_k):

print\_cluster(i)



#### Predicting clusters based on key search words

def show\_recommendations(product):

#print("Cluster ID:")

Y = vectorizer.transform([product])

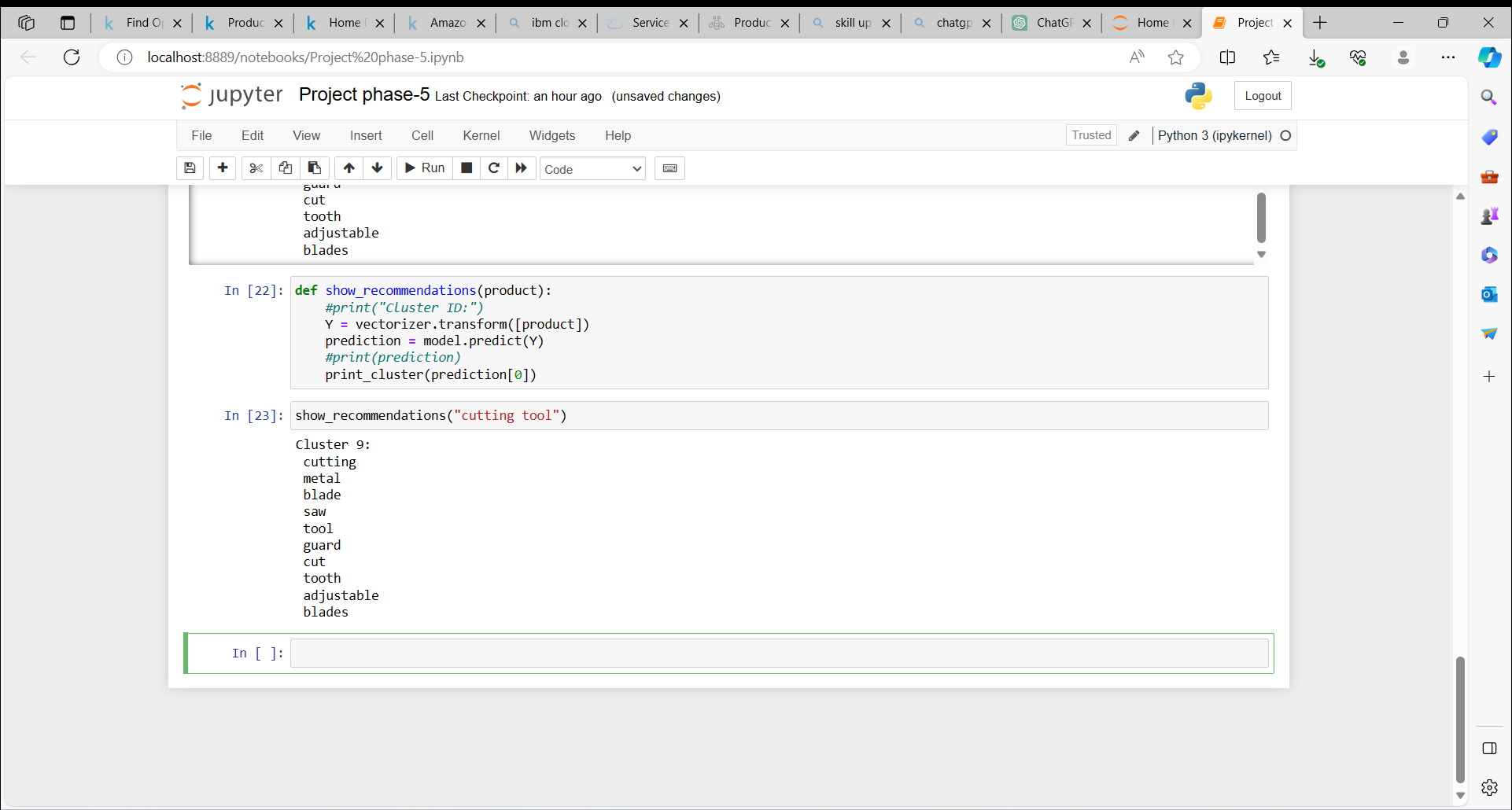
prediction = model.predict(Y)

#print(prediction)

print\_cluster(prediction[0])

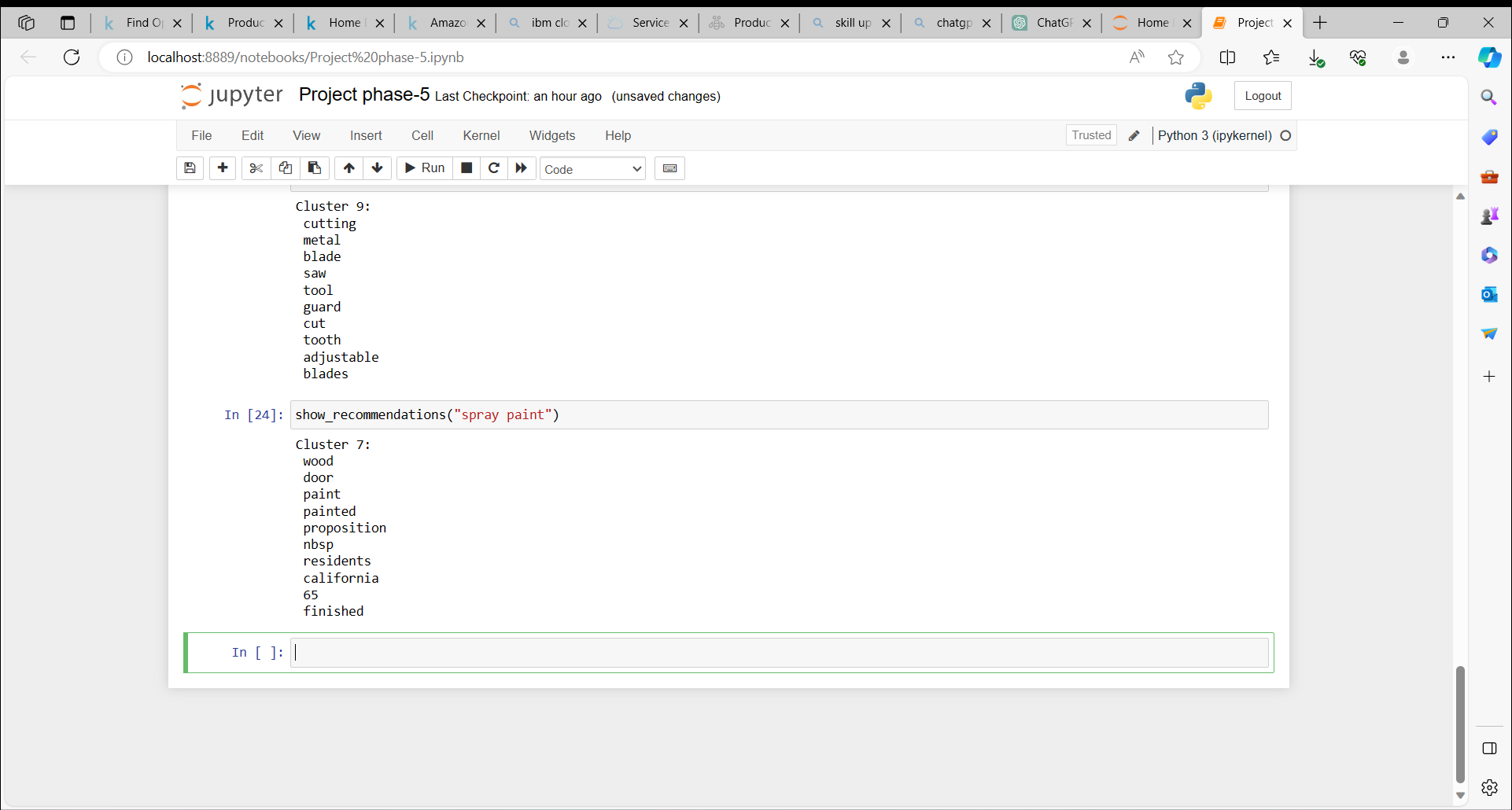
**Keyword :**cutting tool

show\_recommendations("cutting tool")



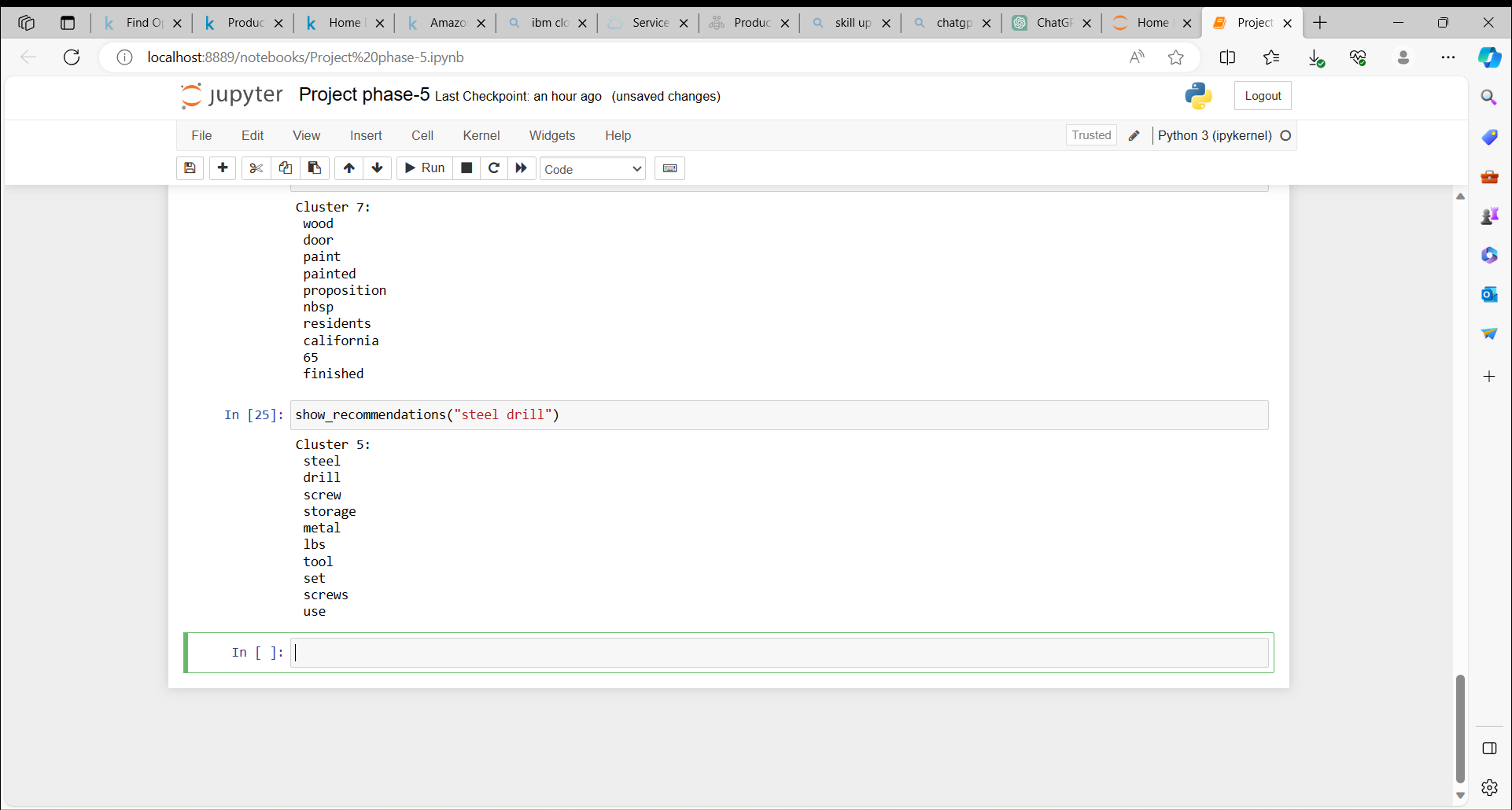
**Keyword :**spray paint

show\_recommendations("spray paint")



**Keyword :**steel drill

show\_recommendations("steel drill")



Once a cluster is identified based on the user's search words, the recommendation system can display items from the corresponding product clusters based on the product descriptions.

To deploy your e-commerce recommendation platform on IBM Cloud Foundry, you'll need to follow these general steps. Keep in mind that specific details may vary depending on your project's setup and the technologies you've used. Here's a high-level guide:

**Prerequisites:**

**1. IBM Cloud Account:** Ensure you have an IBM Cloud account. If you don't have one, sign up at [IBM Cloud] (<https://cloud.ibm.com/>) .

**2. IBM Cloud CLI:** Install the IBM Cloud Command Line Interface (CLI) on your local machine. You can download it from [IBM Cloud CLI] (<https://cloud.ibm.com/docs/cli/reference/ibmcloud?topic=cloud-cli-install-ibmcloud-cli>) .

**3. IBM Cloud Foundry Space:** Set up an IBM Cloud Foundry space where you will deploy your application.

**Deployment Steps:**

**1. Push Your Code to a Git Repository:**

- Make sure your project code is hosted in a Git repository (e.g., GitHub).

**2. Login to IBM Cloud:**

- Open a command line terminal.

- Log in to your IBM Cloud account using the IBM Cloud CLI:

ibmcloud login

**3. Target Your IBM Cloud Foundry Space:**

- Target the IBM Cloud Foundry space where you want to deploy your application:

ibmcloud target --cf

**4. Choose Your Application Type:**

- Depending on your application type (e.g., Node.js, Python, Java), you'll need to specify the appropriate runtime during deployment.

**5. Deploy Your Application:**

- Use the `ibmcloud app push` command to deploy your application. For example, if you have a Node.js application, you can deploy it like this:

ibmcloud app push <your-app-name>

- Replace `<your-app-name>` with the name of your application.

**6. Specify the Manifest File (Optional):**

- If you have a `manifest.yml` file in your project, you can use it to specify application details such as the name, memory, and services. This makes the deployment process more efficient.

- Example manifest.yml:

yaml

applications:

- name: <your-app-name>

memory: 256M

**7. Bind Services (if required):**

- If your application relies on databases, messaging services, or other cloud services, you may need to bind those services to your application. Use the `ibmcloud cf bind-service` command to do this.

**8. Start Your Application:**

- Once deployed, start your application with the `ibmcloud app start` command:

ibmcloud app start <your-app-name>

**9. Access Your Application:**

- After a successful deployment, your application will be accessible via a URL provided by IBM Cloud.

**10. Monitor and Scale:**

- Use the IBM Cloud dashboard to monitor your application's performance and scale as needed.

These are the general steps to deploy an application on IBM Cloud Foundry. Please adapt these steps to your specific project and application requirements. Detailed instructions can vary based on the programming language and frameworks used in your project. Refer to IBM Cloud documentation and guides related to your technology stack for more specific deployment details.

**Conclusion:**

In this project, we set out to create a personalized product recommendation system for an e-commerce platform with the goal of enhancing the user experience and driving sales. By leveraging a diverse set of recommendation algorithms and embracing a design thinking approach, we addressed the challenge of users struggling to find relevant products, ultimately improving the shopping experience and increasing sales potential.

Our primary objectives were to boost user engagement and increase conversion rates by delivering tailored product recommendations. The success of our project was quantified through key performance metrics, including increased click-through rates, prolonged session durations, higher purchase frequency, and improved user satisfaction scores.

Through a design thinking process that involved ideation, user empathy, and feedback mechanisms, we incorporated various recommendation algorithms, including collaborative, content-based, and hybrid models. These algorithms were integrated into our e-commerce platform, which featured personalized homepages, complementary product suggestions, and user profiles. The addition of feedback mechanisms, such as thumbs up/down, comments, surveys, and purchase history reviews, enabled us to continually optimize recommendations and enhance the user experience.

In the development phases, our recommendation system was structured into three components, addressing the diverse needs of both new and returning customers. This approach catered to users whether or not product ratings were available and contributed to an improved customer experience and effective customer acquisition and retention.

The technical implementation details of our project included the use of machine learning algorithms, with model training and deployment facilitated through IBM Cloud Watson Studio. The seamless integration of our recommendation model into the e-commerce platform's backend ensured real-time product recommendations based on user behaviour and preferences.

In conclusion, our personalized product recommendation system for e-commerce successfully met its objectives by enhancing user engagement and driving sales through tailored product recommendations. By combining design thinking, machine learning, and a user-centric approach,

we've provided a platform that not only improves the shopping experience but also supports businesses in acquiring and retaining customers effectively.

The deployment of our platform on IBM Cloud Foundry enables users to access these benefits with ease, making it a valuable addition to the e-commerce landscape. We invite you to explore our GitHub repository, where you can find the project's code, files, and detailed instructions on how to navigate the website, update content, and deploy the platform. We look forward to seeing the impact of this project on the e-commerce industry and user experiences.



01-11-2023