1.
This code mounts Google Drive, reads a CSV file from a specified path in Google Drive into a Pandas DataFrame, and then prints the first few rows of the DataFrame.

2.
This code selects specific columns ('overall', 'reviewTime', 'asin', 'reviewText') from the DataFrame 'df', creates a new DataFrame 'new_df', saves it to a CSV file at the specified output path in Google Drive, and prints the path where the filtered DataFrame is saved along with the DataFrame itself.

3.
This code selects specific columns ('overall', 'reviewerID', 'asin') from the DataFrame 'df', creates a new DataFrame 'last_df', saves it to a CSV file at the specified output path in Google Drive, and prints the path where the DataFrame is saved.

4.
Mounts Google Drive, reads a CSV file ('meta_Electronics.csv') into a Pandas DataFrame named 'df1', and displays the first few rows of the DataFrame. Additionally, the command `!ls` lists the files in the current directory.

5.
This code filters the DataFrame 'df1' to select rows where the 'title' column contains the words 'book', 'books', 'Book', or 'Books' (ignoring case), and keeps only the first occurrence of each unique title. Finally, it displays the resulting DataFrame 'filtered_df'.

6.
This code first extracts unique ASINs (product IDs) from the DataFrame 'filtered_df'. Then, it filters the DataFrame 'new_df' to keep only the rows where the ASINs match those in 'unique_asins'. Next, it merges the filtered rows from both DataFrames based on the ASIN column using an inner join. Finally, it prints the resulting DataFrame 'mergedone', which contains the merged information from both DataFrames.

7.
This code computes various descriptive statistics of the products from the DataFrame 'mergedone', including the number of reviews, average rating score, number of unique products, number of good ratings (ratings >= 3), number of bad ratings (ratings < 3), and the number of reviews corresponding to each rating. Finally, it displays these statistics.

8.
This code defines a series of functions for text preprocessing, including removing HTML tags, accented characters, special characters, and lemmatization. Then, it applies these preprocessing steps to the 'reviewText' column in the DataFrame 'mergedone' using the function 'text_normalizer'. Finally, it displays the resulting DataFrame 'merged_preprocess' after preprocessing.

9.
This code performs various analyses on the preprocessed DataFrame 'merged_preprocess' and visualizes the results:

a. Top 20 most reviewed brands
b. Top 20 least reviewed brands
c. Most positively reviewed product
d. Count of ratings for the product over 5 consecutive years
e. Word Clouds for 'Good' and 'Bad' ratings
f. Distribution of Ratings vs. No. of Reviews (Pie chart)
g. Year with maximum reviews
h. Year with the highest number of customers

The extracted statistics and visualizations are displayed or plotted accordingly.

10.
This code utilizes scikit-learn's `train_test_split` function to split the dataset into training and testing sets. It defines a function `rating_to_class` to convert rating values into classes ('Good', 'Average', 'Bad'). Then, it creates a new column 'Rating Class' based on the 'overall' ratings in the DataFrame 'merged_preprocess'. Afterward, it splits the dataset into input features (X) and the target variable (y), and finally, it splits them into training and testing sets with a 75:25 ratio. The shapes of the resulting train and test sets are displayed.

11
This code defines pipelines for various classification models including Multinomial Naive Bayes, Logistic Regression, Support Vector Machine, Decision Tree, and Random Forest. Each pipeline consists of a TF-IDF vectorizer followed by a classifier. Then, it trains and evaluates each model using the training and testing data, printing out a classification report for each model.

12
This code extracts unique ASINs (product IDs) from the 'asin' column of the DataFrame 'filtered_df'. Then, it filters the DataFrame 'last_df' to keep only the rows where the ASINs match those in 'unique_asins'. Afterward, it merges the filtered rows from both DataFrames based on the 'asin' column using an inner join, resulting in a DataFrame named 'last_merge'. Finally, it displays the resulting DataFrame.

13

This code implements a user-user collaborative filtering recommender system using K-fold cross-validation. It computes the Mean Absolute Error (MAE) for different values of N (number of similar users to consider) using K-fold cross-validation. The process involves splitting the data into training and validation sets, computing the cosine similarity between users, predicting ratings for the validation set based on the weighted average of similar users' ratings, and calculating the MAE for each fold. Finally, it calculates the average MAE across all folds for each value of N and stores the results in the list 'mae_values_user_user'.

14

This code is similar to the previous one but implements an item-item collaborative filtering recommender system using K-fold cross-validation. It calculates the Mean Absolute Error (MAE) for different values of N (number of similar items to consider) using K-fold cross-validation. The process involves splitting the data into training and validation sets, computing the cosine similarity between items, predicting ratings for the validation set based on the weighted average of similar items' ratings, and calculating the MAE for each fold. Finally, it calculates the average MAE across all folds for each value of N and stores the results in the list 'mae_values_item_item'.

15

This code generates a plot showing the Mean Absolute Error (MAE) versus the number of similar users/items (N) for both user-user and item-item collaborative filtering recommender systems. It uses the matplotlib library to create the plot, where the x-axis represents the values of N, and the y-axis represents the corresponding MAE values. The plot displays two lines: one for the user-user recommender system and another for the item-item recommender system. The legend indicates which line corresponds to which recommender system. Finally, the plot is displayed using `plt.show()`.

16

This code calculates the top 10 products by the sum of overall ratings from the DataFrame `last_df`, assuming it contains columns for reviewerID, asin, and overall ratings. It groups the data by the 'asin' column, sums the 'overall' ratings for each product, sorts the resulting sums in descending order, and then selects the top 10 products. Finally, it prints out the top 10 products by user sum ratings.