Market basket analysis is a technique used mostly by retailers to identify which products clients purchase together most frequently. This involves analyzing point of sale (POS) transaction data to identify the correlations between different items according to their co-occurrence in the data.

Using this information, retailers are better able to understand customer purchasing patterns, especially which items customers are likely to purchase together. A well-known example of market basket analysis in action is on the Amazon website. Upon selecting a product, a viewer is taken to the product page which not only includes details about the selected product but also displays a section called “Frequently bought together”.

Though familiar with online retailers, market basket analysis has its roots in brick-and-mortar stores, with decisions like which products to place in close proximity to each other, based on the results of the analysis. For example, if customers are buying steak, how likely are they to buy alcohol (and which type of alcohol)? This information can help guide decisions around product placement, up-selling and cross-selling. It also influences sales promotions, loyalty programs, and pricing strategies like bundling.

While several techniques exist for performing market basket analysis, Association Rules remains one of the most widely used. It identifies strong rules in transaction data by counting the frequency of items that occur together, and finding associations that occur more often than expected. One well-known association rule algorithm used in market basket analysis is the Apriori Algorithm. It works by first finding all frequent attributes in the data set, and then employing association rules based on two metrics, support, and confidence to identify the most important relationships.

In this code-along tutorial, we will focus on how to implement market basket analysis using the apriori algorithm and association rules in Python.

The dataset

To perform our analysis, we'll be using the Online Retail Dataset. This is a dataset containing transnational transactions made on a UK-based online retail store between 01/12/2010 and 09/12/2011. It comprises 541909 rows, with 8 attributes:

InvoiceNo: a unique 6-digit number assigned to each transaction. If this code starts with letter 'C', then the order was cancelled.

letter ‘C’, then the order was cancelled.

StockCode: a unique 5-digit number assigned to each distinct product.

Description: the product name.

Quantity: the number of each product (item) purchased per transaction.

InvoiceDate: the date and time each transaction was completed.

UnitPrice: the product price per unit in pounds sterling.

CustomerID: a unique 5-digit number assigned to each customer.

Country: the name of the country from where the purchase was made.

Importing the libraries

We will begin by importing the relevant python libraries. For our analysis, we’ll be using Pandas for data manipulation, Matplotlib and Seaborn for visualization, and the mlextend library for applying the apriori algorithm and association rules.

Using the info() method, we can check the structure of the dataset, including the number of rows and columns, as well as the names and data type of each column and whether or not they contain null values.

```
#Check the structure of the dataframe
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   InvoiceNo    541909 non-null  objec
 1   StockCode    541909 non-null  objec
 2   Description  540455 non-null  objec
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datet
 5   UnitPrice    541909 non-null  float
 6   CustomerID   406829 non-null  float
 7   Country      541909 non-null  objec
dtypes: datetime64[ns](1), float64(2), i
memory usage: 33.1+ MB
/shared-libs/python3.9/py-core/lib/pytho
  and should_run_async(code)
```

Preparing the data

Our first task in preparing the data is to check the occurrence of null values in the dataset by using the isna() method. By including the sum() method, we can view the total number of null values per column.

```
#Check the occurrence of null values
df.isna().sum()
```

```
/shared-libs/python3.9/py-core/lib/pytho
  and should_run_async(code)
```

```
InvoiceNo            0
StockCode            0
Description       1454
Quantity             0
InvoiceDate          0
UnitPrice            0
CustomerID      135080
Country              0
dtype: int64
```

The dataset contains a total of 136,534 null values, with the vast majority occurring in the CustomerID column. Since we have more than sufficient data for our analysis, we will not bother with trying to impute these missing values, and will simply remove them from the data.

After removing all canceled transactions, we're left with 397,924 rows of data. Our final preparation task is to choose transactions from only one country for our analysis. Before we do so, we can first check how all the transactions are distributed by country with the value_counts() method. We'll limit it to the top 10 countries for easy visibility.

```
#Check the distribution of transactions
top10 = df["Country"].value_counts().he
top10
```

```
/shared-libs/python3.9/py-core/lib/pytho
  and should_run_async(code)
```

```
United Kingdom    354345
Germany             9042
France              8342
EIRE                7238
Spain               2485
Netherlands         2363
Belgium             2031
Switzerland         1842
Portugal            1462
Australia           1185
Name: Country, dtype: int64
```

The UK is clearly number one in terms of the number of transactions. We can visualize this information in a pie chart as well.