```
!pip install apyori
```

```
Collecting apyori
  Downloading apyori-1.1.2.tar.gz (8.6 kB)
  Preparing metadata (setup.py) ... done
Building wheels for collected packages: apyori
  Building wheel for apyori (setup.py) ... done
  Created wheel for apyori: filename=apyori-1.1.2-py3-none-any.whl size=5954 sha256=ba599e95474dafbda548d899062263851e44da8923d1e2e14924
  Stored in directory: /root/.cache/pip/wheels/77/3d/a6/d317a6fb32be58a602b1e8c6b5d6f31f79322da554cad2a5ea
Successfully built apyori
Installing collected packages: apyori
Successfully installed apyori-1.1.2
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from google.colab import files
from apyori import apriori
```

```
uploaded = files.upload()
```

Choose Files Groceries_dataset[1].csv

- **Groceries_dataset[1].csv**(text/csv) - 1103280 bytes, last modified: 7/9/2025 - 100% done

```
Saving Groceries_dataset[1].csv to Groceries_dataset[1] (1).csv
```

```
df = pd.read_csv(next(iter(uploaded)))
```

```
print("Shape:", df.shape)
print(df.describe())
print(df.info())
print("Non-null count:\n", df.notnull().sum())
print("Missing values:\n", df.isna().sum())
df.head()
```

```
Shape: (38765, 3)
       Member_number
count   38765.000000
mean     3003.641868
std      1153.611031
min      1000.000000
25%      2002.000000
50%      3005.000000
75%      4007.000000
max      5000.000000
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 38765 entries, 0 to 38764
Data columns (total 3 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Member_number    38765 non-null  int64
 1   Date             38765 non-null  object
 2   itemDescription  38765 non-null  object
dtypes: int64(1), object(2)
memory usage: 908.7+ KB
None
Non-null count:
 Member_number      38765
Date               38765
itemDescription    38765
dtype: int64
Missing values:
 Member_number      0
Date               0
itemDescription    0
dtype: int64
```

	Member_number	Date	itemDescription
0	1808	21-07-2015	tropical fruit
1	2552	05-01-2015	whole milk
2	2300	19-09-2015	pip fruit
3	1187	12-12-2015	other vegetables
4	3037	01-02-2015	whole milk

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.set_index('Date', inplace=True)
df.index = pd.to_datetime(df.index)
df.head()
```

```
/tmp/ipython-input-8-2081582907.py:2: UserWarning: Parsing dates in %d-%m-%Y format when dayfirst=False (the default) was specified. Pas
  df.index = pd.to_datetime(df.index)
```

Date	Member_number	itemDescription
2015-07-21	1808	tropical fruit
2015-01-05	2552	whole milk
2015-09-19	2300	pip fruit
2015-12-12	1187	other vegetables
2015-02-01	3037	whole milk

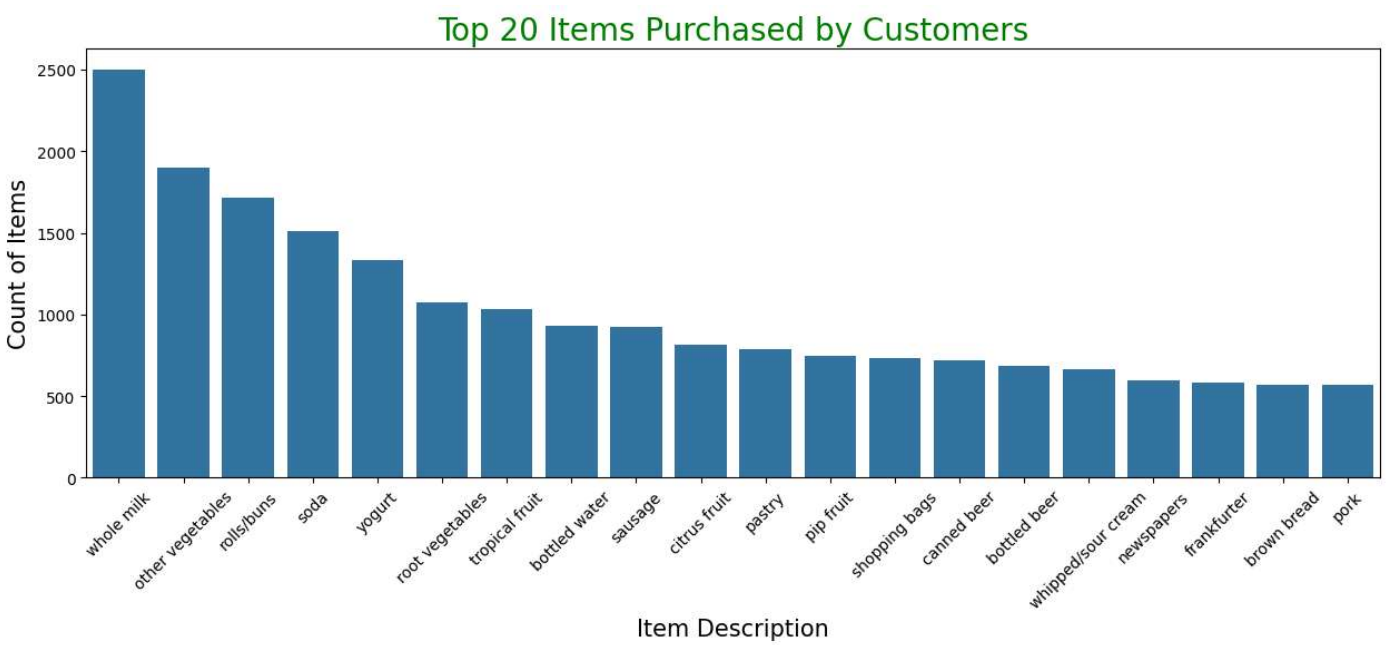
Next steps: Generate code with df | View recommended plots | New interactive sheet

```
total_item = len(df)
total_days = len(np.unique(df.index.date))
total_months = len(np.unique(df.index.year))
print("Total items:", total_item, "| Total days:", total_days, "| Total years:", total_months)
```

```
Total items: 38765 | Total days: 728 | Total years: 2
```

```
plt.figure(figsize=(15, 5))
top_items = df['itemDescription'].value_counts().head(20)
sns.barplot(x=top_items.index, y=top_items.values)
plt.xlabel('Item Description', size=15)
```

```
plt.xticks(rotation=45)
plt.ylabel('Count of Items', size=15)
plt.title('Top 20 Items Purchased by Customers', color='green', size=20)
plt.show()
```




```
df_grouped = df.groupby(['Member_number', 'Date'])['itemDescription'].apply(list)
df_grouped.head()
```

		itemDescription
Member_number	**Date**	
1000	**2014-06-24**	[whole milk, pastry, salty snack]
	2015-03-15	[sausage, whole milk, semi-finished bread, yog...
	2015-05-27	[soda, pickled vegetables]
	2015-07-24	[canned beer, misc. beverages]
	2015-11-25	[sausage, hygiene articles]

dtype: object

```
transactions = df_grouped.values.tolist()
transactions[:10]
```

```
[['whole milk', 'pastry', 'salty snack'],
 ['sausage', 'whole milk', 'semi-finished bread', 'yogurt'],
 ['soda', 'pickled vegetables'],
 ['canned beer', 'misc. beverages'],
 ['sausage', 'hygiene articles'],
 ['sausage', 'whole milk', 'rolls/buns'],
 ['whole milk', 'soda'],
 ['frankfurter', 'soda', 'whipped/sour cream'],
 ['beef', 'white bread'],
 ['frankfurter', 'curd']]
```

```
rules = apriori(transactions, min_support=0.0003, min_confidence=0.05, min_lift=2, min_length=2)
results = list(rules)
```

```
def inspect(results):
    lhs = [list(rule.ordered_statistics[0].items_base)[0] for rule in results]
    rhs = [list(rule.ordered_statistics[0].items_add)[0] for rule in results]
    supports = [rule.support for rule in results]
```

```
    confidences = [rule.ordered_statistics[0].confidence for rule in results]
    lifts = [rule.ordered_statistics[0].lift for rule in results]
    return list(zip(lhs, rhs, supports, confidences, lifts))


if results:
    ordered_results = pd.DataFrame(inspect(results), columns=['Left Hand Side', 'Right Hand Side', 'Support', 'Confidence', 'Lift'])
    display(ordered_results)
else:
    print("No strong association rules found. Try adjusting support or confidence.")
```

	Left Hand Side	Right Hand Side	Support	Confidence	Lift
0	artif. sweetener	soda	0.000468	0.241379	2.485725
1	condensed milk	berries	0.000334	0.051020	2.341774
2	brandy	whole milk	0.000869	0.342105	2.166281
3	sweet spreads	butter	0.000334	0.073529	2.087705
4	liver loaf	canned beer	0.000401	0.120000	2.557778
...	...	...	...	...	...
99	whipped/sour cream	yogurt	0.000601	0.204545	2.381800