```
In [1]: import pandas as pd
```

1. Load Data

```
In [2]: df = pd.read_excel('http://people.bu.edu/kalathur/datasets/OnlineRetail.xlsx', sheet_name='Online Retail')
```

```
In [3]: df.shape
```

```
Out[3]: (541909, 8)
```

```
In [4]: df.head()
```

Out[4]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

```
In [5]: df = df.loc[df['Quantity'] > 0]
```

```
In [6]: df.shape
```

```
Out[6]: (531285, 8)
```

2. Data Preparation

- Handle NaNs in CustomerID field

In [7]:
```
df['CustomerID'].describe()
```

Out[7]:
```
count    397924.000000
mean      15294.315171
std        1713.169877
min       12346.000000
25%       13969.000000
50%       15159.000000
75%       16795.000000
max       18287.000000
Name: CustomerID, dtype: float64
```

In [8]:
```
df['CustomerID'].isna().sum()
```

Out[8]:
```
133361
```

In [9]:
```
df.loc[df['CustomerID'].isna()].head()
```

Out[9]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
622	536414	22139	NaN	56	2010-12-01 11:52:00	0.00	NaN	United Kingdom
1443	536544	21773	DECORATIVE ROSE BATHROOM BOTTLE	1	2010-12-01 14:32:00	2.51	NaN	United Kingdom
1444	536544	21774	DECORATIVE CATS BATHROOM BOTTLE	2	2010-12-01 14:32:00	2.51	NaN	United Kingdom
1445	536544	21786	POLKADOT RAIN HAT	4	2010-12-01 14:32:00	0.85	NaN	United Kingdom
1446	536544	21787	RAIN PONCHO RETROSPOT	2	2010-12-01 14:32:00	1.66	NaN	United Kingdom

In [10]:
```
df.shape
```

Out[10]:
```
(531285, 8)
```

In [11]:
```
df = df.dropna(subset=['CustomerID'])
```

In [12]:
```
df.shape
```

Out[12]:
```
(397924, 8)
```

In [13]:
```
df.head()
```

Out[13]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

- Customer-Item Matrix

In [14]:
```
customer_item_matrix = df.pivot_table(
    index='CustomerID',
    columns='StockCode',
    values='Quantity',
    aggfunc='sum'
)
```

In [15]:
```
customer_item_matrix.loc[12481:].head()
```

Out[15]:

StockCode	10002	10080	10120	10125	10133	10135	11001	15030	15034	15036	...	90214V	90214W	90214Y	90214Z	BANK CHARGES	C2	DOT	M
CustomerID																			
12481.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	36.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
12483.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
12484.0	NaN	NaN	NaN	NaN	NaN	NaN	16.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
12488.0	NaN	NaN	NaN	NaN	NaN	10.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
12489.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

5 rows × 3665 columns

In [16]:
```
customer_item_matrix.shape
```

Out[16]:
```
(4339, 3665)
```

In [17]:
```
df['StockCode'].nunique()
```

Out[17]:
```
3665
```

In [18]:
```
df['CustomerID'].nunique()
```

Out[18]:
```
4339
```

In [19]:
```
customer_item_matrix.loc[12348.0].sum()
```

Out[19]:
```
2341.0
```

In [20]:
```
customer_item_matrix = customer_item_matrix.applymap(lambda x: 1 if x > 0 else 0)
```

In [21]:
```
customer_item_matrix.loc[12481:].head()
```

Out[21]:

StockCode	10002	10080	10120	10125	10133	10135	11001	15030	15034	15036	...	90214V	90214W	90214Y	90214Z	BANK CHARGES	C2	DOT	M	F
CustomerID																				
12481.0	0	0	0	0	0	0	0	0	0	1	...	0	0	0	0	0	0	0	0	
12483.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	
12484.0	0	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	
12488.0	0	0	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	
12489.0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	

5 rows × 3665 columns

3. Collaborative Filtering

In [22]:
```
from sklearn.metrics.pairwise import cosine_similarity
```

3.1. User-based Collaborative Filtering

- User-to-User Similarity Matrix

In [23]:
```
user_user_sim_matrix = pd.DataFrame(
    cosine_similarity(customer_item_matrix)
)
```

In [24]:
```
user_user_sim_matrix.head()
```

Out[24]:

	0	1	2	3	4	5	6	7	8	9	...	4329	4330	4331	4332	4333	4334	433
0	1.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	...	0.0	0.000000	0.000000	0.0	0.000000	0.000000	0
1	0.0	1.000000	0.063022	0.046130	0.047795	0.038484	0.0	0.025876	0.136641	0.094742	...	0.0	0.029709	0.052668	0.0	0.032844	0.062318	0
2	0.0	0.063022	1.000000	0.024953	0.051709	0.027756	0.0	0.027995	0.118262	0.146427	...	0.0	0.064282	0.113961	0.0	0.000000	0.000000	0
3	0.0	0.046130	0.024953	1.000000	0.056773	0.137137	0.0	0.030737	0.032461	0.144692	...	0.0	0.105868	0.000000	0.0	0.039014	0.000000	0
4	0.0	0.047795	0.051709	0.056773	1.000000	0.031575	0.0	0.000000	0.000000	0.033315	...	0.0	0.000000	0.000000	0.0	0.000000	0.000000	0

5 rows × 4339 columns

In [25]:
```
user_user_sim_matrix.columns = customer_item_matrix.index

user_user_sim_matrix['CustomerID'] = customer_item_matrix.index
user_user_sim_matrix = user_user_sim_matrix.set_index('CustomerID')
```

In [26]:
```
user_user_sim_matrix.head()
```

Out[26]:

CustomerID	12346.0	12347.0	12348.0	12349.0	12350.0	12352.0	12353.0	12354.0	12355.0	12356.0	...	18273.0	18274.0	18276.0	18277.0
CustomerID															
12346.0	1.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.000000	0.000000	...	0.0	0.000000	0.000000	0.0
12347.0	0.0	1.000000	0.063022	0.046130	0.047795	0.038484	0.0	0.025876	0.136641	0.094742	...	0.0	0.029709	0.052668	0.0
12348.0	0.0	0.063022	1.000000	0.024953	0.051709	0.027756	0.0	0.027995	0.118262	0.146427	...	0.0	0.064282	0.113961	0.0
12349.0	0.0	0.046130	0.024953	1.000000	0.056773	0.137137	0.0	0.030737	0.032461	0.144692	...	0.0	0.105868	0.000000	0.0
12350.0	0.0	0.047795	0.051709	0.056773	1.000000	0.031575	0.0	0.000000	0.000000	0.033315	...	0.0	0.000000	0.000000	0.0

5 rows × 4339 columns

- Making Recommendations

In [27]:
```
user_user_sim_matrix.loc[12350.0].sort_values(ascending=False)
```

Out[27]:
```
CustomerID
12350.0    1.000000
17935.0    0.183340
12414.0    0.181902
12652.0    0.175035
16692.0    0.171499
             ...
15953.0    0.000000
15952.0    0.000000
15951.0    0.000000
15950.0    0.000000
12346.0    0.000000
Name: 12350.0, Length: 4339, dtype: float64
```

In [28]:
```
items_bought_by_A = set(customer_item_matrix.loc[12350.0].iloc[
    customer_item_matrix.loc[12350.0].nonzero()
].index)
items_bought_by_A
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/ipykernel_launcher.py:2: FutureWa
rning: Series.nonzero() is deprecated and will be removed in a future version.Use Series.to_numpy().nonzero() i
nstead
```

Out[28]:
```
{20615,
 20652,
 21171,
 21832,
 21864,
 21866,
 21908,
 21915,
 22348,
 22412,
 22551,
 22557,
 22620,
 '79066K',
 '79191C',
```

In [29]:
```
items_bought_by_B = set(customer_item_matrix.loc[17935.0].iloc[
    customer_item_matrix.loc[17935.0].nonzero()
].index)
items_bought_by_B
```

```
/Library/Frameworks/Python.framework/Versions/3.7/lib/python3.7/site-packages/ipykernel_launcher.py:2: FutureWa
rning: Series.nonzero() is deprecated and will be removed in a future version.Use Series.to_numpy().nonzero() i
nstead
```

Out[29]:
```
{20657,
 20659,
 20828,
 20856,
 21051,
 21866,
 21867,
 22208,
 22209,
 22210,
 22211,
 22449,
 22450,
 22551,
 22553,
```

In [30]:
```
items_to_recommend_to_B = items_bought_by_A - items_bought_by_B
```

In [31]:
```
items_to_recommend_to_B
```

Out[31]:
```
{20615,
 20652,
 21171,
 21832,
 21864,
 21908,
 21915,
 22348,
 22412,
 22620,
 '79066K',
 '79191C',
 '84086C'}
```

In [32]:
```
df.loc[
    df['StockCode'].isin(items_to_recommend_to_B),
    ['StockCode', 'Description']
].drop_duplicates().set_index('StockCode')
```

Out[32]:

StockCode	Description
21832	CHOCOLATE CALCULATOR
21915	RED HARMONICA IN BOX
22620	4 TRADITIONAL SPINNING TOPS
79066K	RETRO MOD TRAY
21864	UNION JACK FLAG PASSPORT COVER
79191C	RETRO PLASTIC ELEPHANT TRAY
21908	CHOCOLATE THIS WAY METAL SIGN
20615	BLUE POLKADOT PASSPORT COVER
20652	BLUE POLKADOT LUGGAGE TAG
22348	TEA BAG PLATE RED RETROSPOT
22412	METAL SIGN NEIGHBOURHOOD WITCH
21171	BATHROOM METAL SIGN
84086C	PINK/PURPLE RETRO RADIO

3.2. Item-based Collaborative Filtering

- Item-to-Item Similarity Matrix

In [33]:
```
item_item_sim_matrix = pd.DataFrame(cosine_similarity(customer_item_matrix.T))
```

In [34]:
```
item_item_sim_matrix.columns = customer_item_matrix.T.index

item_item_sim_matrix['StockCode'] = customer_item_matrix.T.index
item_item_sim_matrix = item_item_sim_matrix.set_index('StockCode')
```

In [35]:
```
item_item_sim_matrix
```

Out[35]:

StockCode	10002	10080	10120	10125	10133	10135	11001	15030	15034	15036	...	90214V	90214W	90214Y	90214Z
StockCode															
10002	1.000000	0.000000	0.094868	0.090351	0.062932	0.098907	0.095346	0.047673	0.075593	0.090815	...	0.000000	0.0	0.000000	0.0
10080	0.000000	1.000000	0.000000	0.032774	0.045655	0.047836	0.000000	0.000000	0.082261	0.049413	...	0.000000	0.0	0.000000	0.0
10120	0.094868	0.000000	1.000000	0.057143	0.059702	0.041703	0.060302	0.060302	0.095618	0.028718	...	0.000000	0.0	0.000000	0.0
10125	0.090351	0.032774	0.057143	1.000000	0.042644	0.044682	0.043073	0.000000	0.051224	0.030770	...	0.000000	0.0	0.000000	0.0
10133	0.062932	0.045655	0.059702	0.042644	1.000000	0.280097	0.045002	0.060003	0.071358	0.057152	...	0.000000	0.0	0.000000	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
C2	0.029361	0.000000	0.000000	0.000000	0.036955	0.019360	0.055989	0.000000	0.000000	0.039996	...	0.000000	0.0	0.000000	0.0
DOT	0.000000	0.000000	0.000000	0.000000	0.000000	0.104257	0.150756	0.000000	0.000000	0.000000	...	0.000000	0.0	0.000000	0.0
M	0.066915	0.016182	0.070535	0.070535	0.070185	0.066184	0.106335	0.063801	0.059013	0.086089	...	0.049875	0.0	0.040723	0.0

- Making Recommendations

In [36]:
```
top_10_similar_items = list(
    item_item_sim_matrix\
        .loc[23166]\
        .sort_values(ascending=False)\
        .iloc[:10]\
    .index
)
```

In [37]:
```
top_10_similar_items
```

Out[37]:
```
[23166, 23165, 23167, 22993, 23307, 22722, 22720, 22666, 23243, 22961]
```

In [38]:
```
df.loc[
    df['StockCode'].isin(top_10_similar_items),
    ['StockCode', 'Description']
].drop_duplicates().set_index('StockCode').loc[top_10_similar_items]
```

Out[38]:

StockCode	Description
23166	MEDIUM CERAMIC TOP STORAGE JAR
23165	LARGE CERAMIC TOP STORAGE JAR
23167	SMALL CERAMIC TOP STORAGE JAR
22993	SET OF 4 PANTRY JELLY MOULDS
23307	SET OF 60 PANTRY DESIGN CAKE CASES
22722	SET OF 6 SPICE TINS PANTRY DESIGN
22720	SET OF 3 CAKE TINS PANTRY DESIGN
22666	RECIPE BOX PANTRY YELLOW DESIGN
23243	SET OF TEA COFFEE SUGAR TINS PANTRY
22961	JAM MAKING SET PRINTED

In [ ]: