STA6714 ASSIGNMENT-7

ONLINE RETAIL DATASET:

The Online Retail dataset is a publicly available set of sales transactions for an online retailer with no physical stores that took place between December 1, 2010, and December 9, 2011, in the UK. There are 8,000 consumers and around 1.1 million transactions in the dataset. 5,900 different things that the shop sells are included in the transactions. The information in the data consists of details like the customer ID, invoice number, invoice date, stock code (item code), description, quantity, price, and country. The dataset is commonly utilized in the field of retail analytics for market basket analysis. It may be used to investigate client purchasing trends, carry out customer segmentation, and forecast upcoming sales.

STEPS FOLLOWED:

1. Loaded the dataset from excel into a dataframe.

2. Clean and preprocess the data to ensure it is in a suitable format for modeling. This includes handling missing values, removing duplicates, and transforming the data into a suitable format.

EDA & DATA PREPROCESSING

```
In [3]: 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  object
 1   StockCode    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

```
In [4]: df.isnull().sum()
```

```
Out[4]: InvoiceNo          0
StockCode          0
Description     1454
Quantity           0
InvoiceDate        0
UnitPrice          0
CustomerID    135080
Country            0
dtype: int64
```

```
In [5]: df.dropna(inplace=True)
```

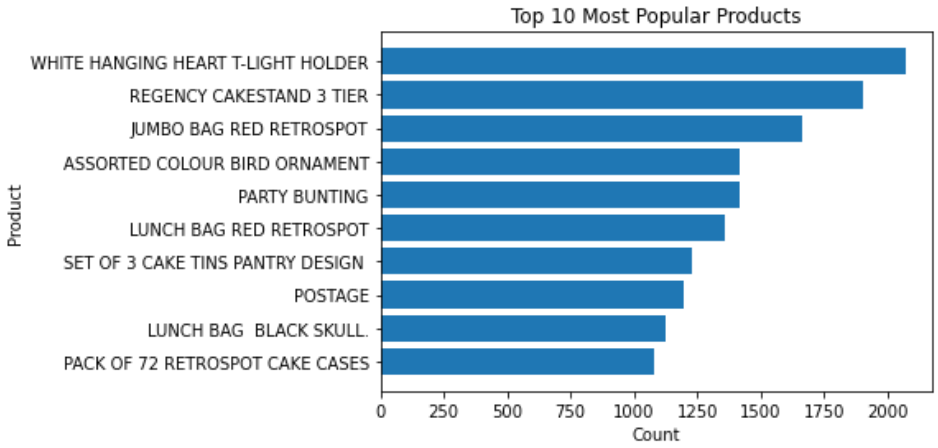
```
In [6]: df.head()
```

Out[6]:

	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 [7]: print(f"Number of unique transactions: {df['InvoiceNo'].nunique()}")
```

```
Number of unique transactions: 22190
```



- From the dataset, the above diagram shows the top 10 popular products.
- The dataset shows that the UK is having the greatest number of transactions.
- There are a total of 22190 number of transactions.

3. Group the items into categories to make the recommendations more meaningful.
   For this, I extracted the nouns from the item description and then created categories to make it easy for the recommendation. Then used the levenshtein_similarity to find the similar words. Used the fuzzywuzzy package to get the closest match for an item.

Extract the Nouns from the item description to make it easy for the modelling

In [8]:
```
nlp = spacy.load('en_core_web_sm')

def extract_nouns(text):
    doc = nlp(text)
    nouns = []
    for token in doc:
        if token.pos_ in ['NOUN', 'PROPN']:
            nouns.append(token.text)
    return nouns

unique_desc_df = df['Description'].unique()

desc_nouns_dict = {}
for desc in tqdm(unique_desc_df, desc='Extracting nouns', total=len(unique_desc_df)):
    desc_nouns_dict[desc] = extract_nouns(desc)

tqdm.pandas(desc='Mapping nouns')
df['nouns'] = df['Description'].progress_map(lambda desc: desc_nouns_dict[desc])
```

In [11]:
```
nlp = spacy.load('en_core_web_sm')

def create_category_name(items):
    item_text = " ".join(items)
    doc = nlp(item_text)
    word_freq = {}
    for token in doc:
        if token.is_alpha and not token.is_stop and len(token.text) > 2:
            if token.text.lower() in word_freq:
                word_freq[token.text.lower()] += 1
            else:
                word_freq[token.text.lower()] = 1
    top_words = sorted(word_freq, key=word_freq.get, reverse=True)[:3]
    category_name = " ".join(top_words)
    return category_name
```

In [12]:
```
def levenshtein_similarity(s1, s2):
    return fuzz.ratio(s1.lower(), s2.lower())
```

4. Generate recommendations for each invoice and store them in a new DataFrame.
   To generate the recommendation, get_recommendation() function is defined and each invoice is passed to the function which recommends an item for each invoice.

In [23]:
```
def get_recommendation(invoice_no):
    basket = get_invoice_items(invoice_no, get='nouns')
    basket = [" ".join(extract_nouns(text)) for text in basket]

    basket_category = create_category_name(basket)

    matches = process.extract(basket_category, df["Category"].unique(), scorer=fuzz.token_sort_ratio)
    matches_sorted = sorted(matches, key=lambda x: x[1], reverse=True)

    for closest_match, _ in matches_sorted:
        similar_items = get_df_items(columnname="Category", columnID=closest_match).sort_values(by='Quantity', ascending=False)

        similar_items = similar_items[~similar_items['Description'].isin(basket)]

        if len(similar_items) > 0:
            recommendation = np.random.choice(similar_items['Description'].unique()[:3])
            return recommendation

    return "No recommendation found."
```

In [24]:
```
def get_basket_rec(basket):
    basket_category = create_category_name(basket)

    matches = process.extract(basket_category, df["Category"].unique(), scorer=fuzz.token_sort_ratio)
    matches_sorted = sorted(matches, key=lambda x: x[1], reverse=True)

    for closest_match, _ in matches_sorted:
        similar_items = get_df_items(columnname="Category", columnID=closest_match).sort_values(by='Quantity', ascending=False)

        similar_items = similar_items[~similar_items['Description'].isin(basket)]

        if len(similar_items) > 0:
            recommendation = np.random.choice(similar_items['Description'].unique()[:3])
            return recommendation
```

5. Create the recommendation dataset and summary table and store them in CSV Files.

LOADING THE RESULTS INTO CSV FILES

In [27]:
```
df[["InvoiceNo", "Recommendation"]].to_csv('recommendations_dp.csv', index=False)
```

In [28]:
```
recommendation_dataset = df.drop_duplicates(subset=["InvoiceNo"])
recommendation_dataset[["InvoiceNo", "Recommendation"]].to_csv('recommendations_dp.csv', index=False)
```

In [29]:
```
recommendation_dataset
```

Out[29]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	nouns	Category	Recommendation
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	WHITE HANGING HEART T LIGHT HOLDER	pink heart set	WOODEN PICTURE FRAME WHITE FINISH
7	536366	22633	HAND WARMER UNION JACK	6	2010-12-01 08:28:00	1.85	17850.0	United Kingdom	HAND WARMER UNION JACK	red hand warmer	BALLOON WATER BOMB PACK OF 35
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	2010-12-01 08:34:00	1.69	13047.0	United Kingdom	ASSORTED COLOUR BIRD ORNAMENT	assorted colour bird	SET3 BOOK BOX GREEN GINGHAM FLOWER

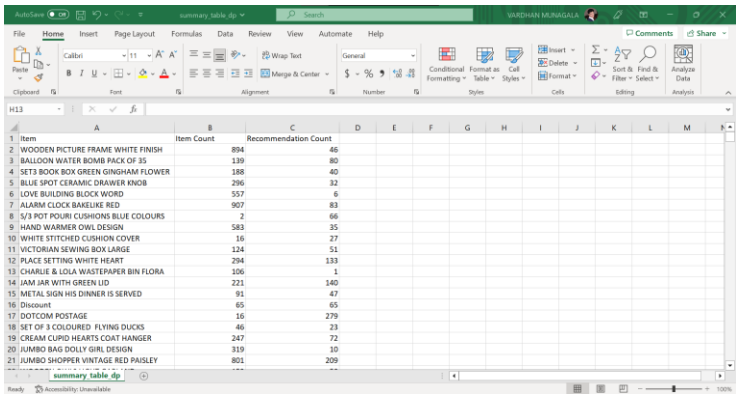
In [36]:
```
summary_table = pd.DataFrame(columns=['Item', 'Item Count', 'Recommendation Count'])

unique_recs = df["Recommendation"].unique()
for item in tqdm(unique_recs):
    invoice_count = df[df['Description'] == item]['InvoiceNo'].nunique()
    recommendation_count = recommendation_dataset[recommendation_dataset['Recommendation'] == item]['InvoiceNo'].count()
    summary_table = pd.concat([summary_table, pd.DataFrame([[item, invoice_count, recommendation_count]], columns=['Item', 'Item

summary_table.to_csv('summary_table_dp.csv',index= False)
```

```
100%|██████████| 513/513 [00:15<00:00, 33.47it/s]
```

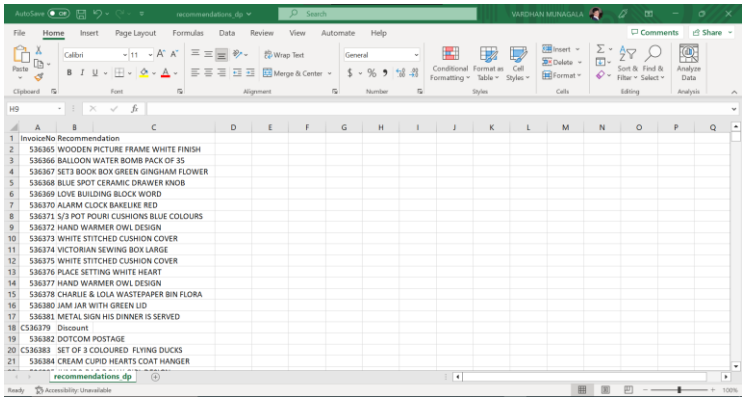
In [38]:
```
summary_table
```

Out[38]:

	Item	Item Count	Recommendation Count
0	WOODEN PICTURE FRAME WHITE FINISH	894	46
0	BALLOON WATER BOMB PACK OF 35	139	80
0	SET3 BOOK BOX GREEN GINGHAM FLOWER	188	40
0	BLUE SPOT CERAMIC DRAWER KNOB	296	32
0	LOVE BUILDING BLOCK WORD	557	6

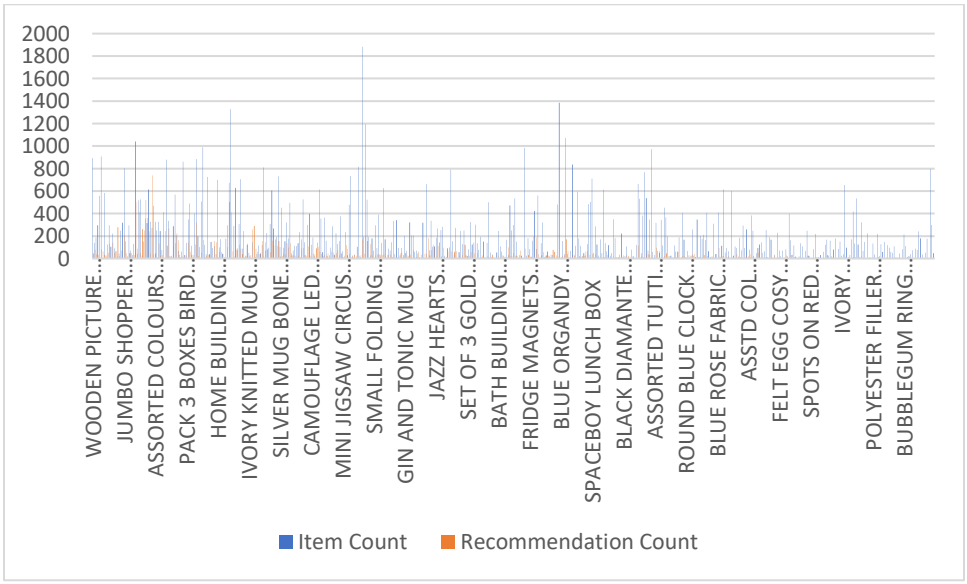


Item	Item Count	Recommendation Count
WOODEN PICTURE FRAME WHITE FINISH	894	46
BALLOON WATER BOMB PACK OF 35	139	80
SET3 BOOK BOX GREEN GINGHAM FLOWER	188	40
BLUE SPOT CERAMIC DRAWER KNOB	296	32
LOVE BUILDING BLOCK WORD	557	6
ALARM CLOCK BAKELIKE RED	907	83
S/3 POT POURI CUSHIONS BLUE COLOURS	2	66
HAND WARMER OWL DESIGN	583	35
WHITE STITCHED CUSHION COVER	16	27
VICTORIAN SEWING BOX LARGE	124	51
PLACE SETTING WHITE HEART	294	133
CHARLIE & LOLA WASTEPAPER BIN FLORA	106	1
JAM JAR WITH GREEN LID	221	140
METAL SIGN HIS DINNER IS SERVED	91	47
Discount	65	65
DOTCOM POSTAGE	16	279
SET OF 3 COLOURED FLYING DUCKS	46	23
CREAM CUPID HEARTS COAT HANGER	247	72
JUMBO BAG DOLLY GIRL DESIGN	319	10
JUMBO SHOPPER VINTAGE RED PAISLEY	801	209



InvoiceNo	Recommendation
536365	WOODEN PICTURE FRAME WHITE FINISH
536366	BALLOON WATER BOMB PACK OF 35
536367	SET3 BOOK BOX GREEN GINGHAM FLOWER
536368	BLUE SPOT CERAMIC DRAWER KNOB
536369	LOVE BUILDING BLOCK WORD
536370	ALARM CLOCK BAKELIKE RED
536371	S/3 POT POURI CUSHIONS BLUE COLOURS
536372	HAND WARMER OWL DESIGN
536373	WHITE STITCHED CUSHION COVER
536374	VICTORIAN SEWING BOX LARGE
536375	WHITE STITCHED CUSHION COVER
536376	PLACE SETTING WHITE HEART
536377	HAND WARMER OWL DESIGN
536378	CHARLIE & LOLA WASTEPAPER BIN FLORA
536380	JAM JAR WITH GREEN LID
536381	METAL SIGN HIS DINNER IS SERVED
C536379	Discount
536382	DOTCOM POSTAGE
C536383	SET OF 3 COLOURED FLYING DUCKS
536384	CREAM CUPID HEARTS COAT HANGER

Summary Table Visualization:



The above diagram shows the item count and the recommendation count of each item in the summary table.