# Shopping Cart Items Recommendation

## Big Data Analytics Mini Project

In [1]:

```
%matplotlib inline
import pandas as pd
import numpy as np
import random
import sys
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.metrics.pairwise import cosine_similarity
```

In [2]:

```
data_products_prior = pd.read_csv("C:/Users/Harsh Patel/Desktop/Big Data Mini Project/order
ordersDF = pd.read_csv("C:/Users/Harsh Patel/Desktop/Big Data Mini Project/orders.csv")
productsDF = pd.read_csv("C:/Users/Harsh Patel/Desktop/Big Data Mini Project/products.csv")
order_products_train = pd.read_csv("C:/Users/Harsh Patel/Desktop/Big Data Mini Project/orde
order_products_prior = pd.read_csv("C:/Users/Harsh Patel/Desktop/Big Data Mini Project/orde
```

In [3]:

```
print("The order_products_train size is : ", order_products_train.shape)
print("The order_products_prior size is : ", order_products_prior.shape)
```

```
The order_products_train size is :  (1384617, 4)
The order_products_prior size is :  (32434489, 4)
```

In [4]:

```
order_products_train.head(2)
```

Out[4]:

| | order_id | product_id | add_to_cart_order | reordered |
|---|---|---|---|---|
| **0** | 1 | 49302 | 1 | 1 |
| **1** | 1 | 11109 | 2 | 1 |

In [5]:

```
order_products_prior.head(2)
```

Out[5]:

| | order_id | product_id | add_to_cart_order | reordered |
|---|---|---|---|---|
| **0** | 2 | 33120 | 1 | 1 |
| **1** | 2 | 28985 | 2 | 1 |

In [6]:

```
order_products_all = pd.concat([order_products_train, order_products_prior], axis=0)

print("The order_products_all size is : ", order_products_all.shape)
```

```
The order_products_all size is :  (33819106, 4)
```

In [7]:

```
order_products_all.head(5)
```

Out[7]:

| | order_id | product_id | add_to_cart_order | reordered |
|---|---|---|---|---|
| **0** | 1 | 49302 | 1 | 1 |
| **1** | 1 | 11109 | 2 | 1 |
| **2** | 1 | 10246 | 3 | 0 |
| **3** | 1 | 49683 | 4 | 0 |
| **4** | 1 | 43633 | 5 | 1 |

In [8]:

```
total = order_products_all.isnull().sum().sort_values(ascending=False)
percent = (order_products_all.isnull().sum()/order_products_all.isnull().count()).sort_valu
missing_data = pd.concat([total, percent], axis=1, keys=['Total Missing', 'Percent'])
missing_data
```

Out[8]:

| | Total Missing | Percent |
|---|---|---|
| **reordered** | 0 | 0.0 |
| **add_to_cart_order** | 0 | 0.0 |
| **product_id** | 0 | 0.0 |
| **order_id** | 0 | 0.0 |

In [9]:

```
#the most ordered products
grouped = order_products_all.groupby("product_id")["reordered"].aggregate({'Total_reorders'
grouped = pd.merge(grouped, productsDF[['product_id', 'product_name']], how='left', on=['pr
grouped = grouped.sort_values(by='Total_reorders', ascending=False)[:10]
grouped
```

```
C:\Anaconda\lib\site-packages\ipykernel_launcher.py:2: FutureWarning: using
a dict on a Series for aggregation
is deprecated and will be removed in a future version
```

Out[9]:

| | product_id | Total_reorders | product_name |
|---|---|---|---|
| **24849** | 24852 | 491291 | Banana |
| **13173** | 13176 | 394930 | Bag of Organic Bananas |
| **21134** | 21137 | 275577 | Organic Strawberries |
| **21900** | 21903 | 251705 | Organic Baby Spinach |
| **47205** | 47209 | 220877 | Organic Hass Avocado |
| **47762** | 47766 | 184224 | Organic Avocado |
| **47622** | 47626 | 160792 | Large Lemon |
| **16794** | 16797 | 149445 | Strawberries |
| **26206** | 26209 | 146660 | Limes |
| **27842** | 27845 | 142813 | Organic Whole Milk |

In [10]:

```
grouped = ordersDF.groupby("eval_set")["order_id"].aggregate({'Total_orders': 'count'}).res
grouped['Ratio'] = grouped["Total_orders"].apply(lambda x: x /grouped['Total_orders'].sum()
grouped
```

```
C:\Anaconda\lib\site-packages\ipykernel_launcher.py:1: FutureWarning: using
a dict on a Series for aggregation
is deprecated and will be removed in a future version
  """Entry point for launching an IPython kernel.
```

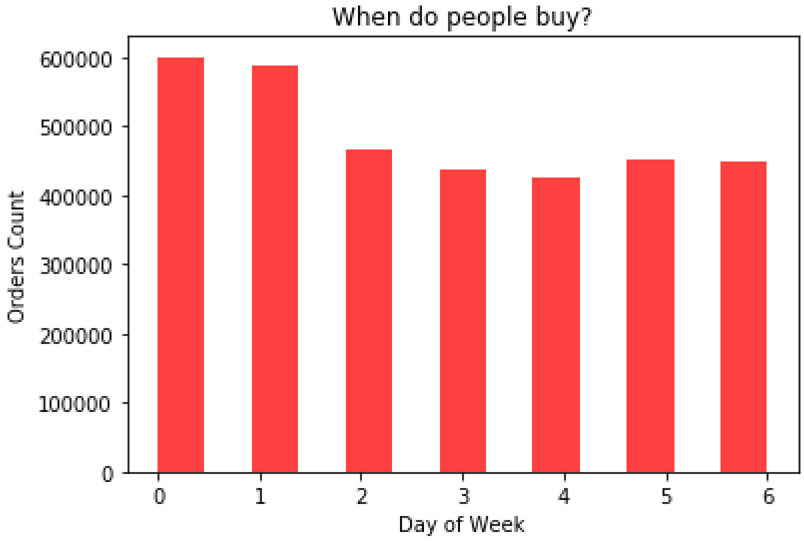
Out[10]:

| | eval_set | Total_orders | Ratio |
|---|---|---|---|
| **0** | prior | 3214874 | 0.939724 |
| **1** | test | 75000 | 0.021923 |
| **2** | train | 131209 | 0.038353 |

# Data Visualization

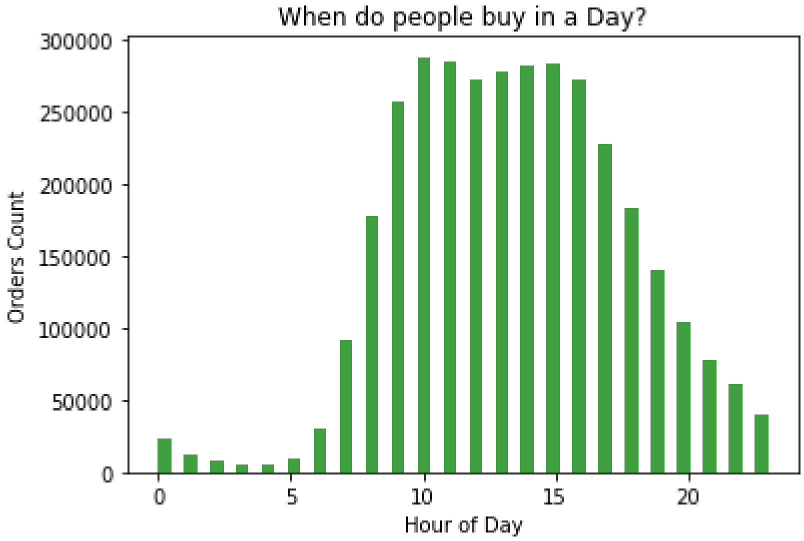
In [11]:

```
#unique DOW values
ordersDF.order_dow.unique()
#what is the frequency of the orders according to days
n, bins, patches = plt.hist(ordersDF.order_dow, 13, facecolor="red", alpha=.75, align='mid'
plt.xlabel("Day of Week")
plt.ylabel("Orders Count")
plt.title("When do people buy?")
plt.show()
```



In [12]:

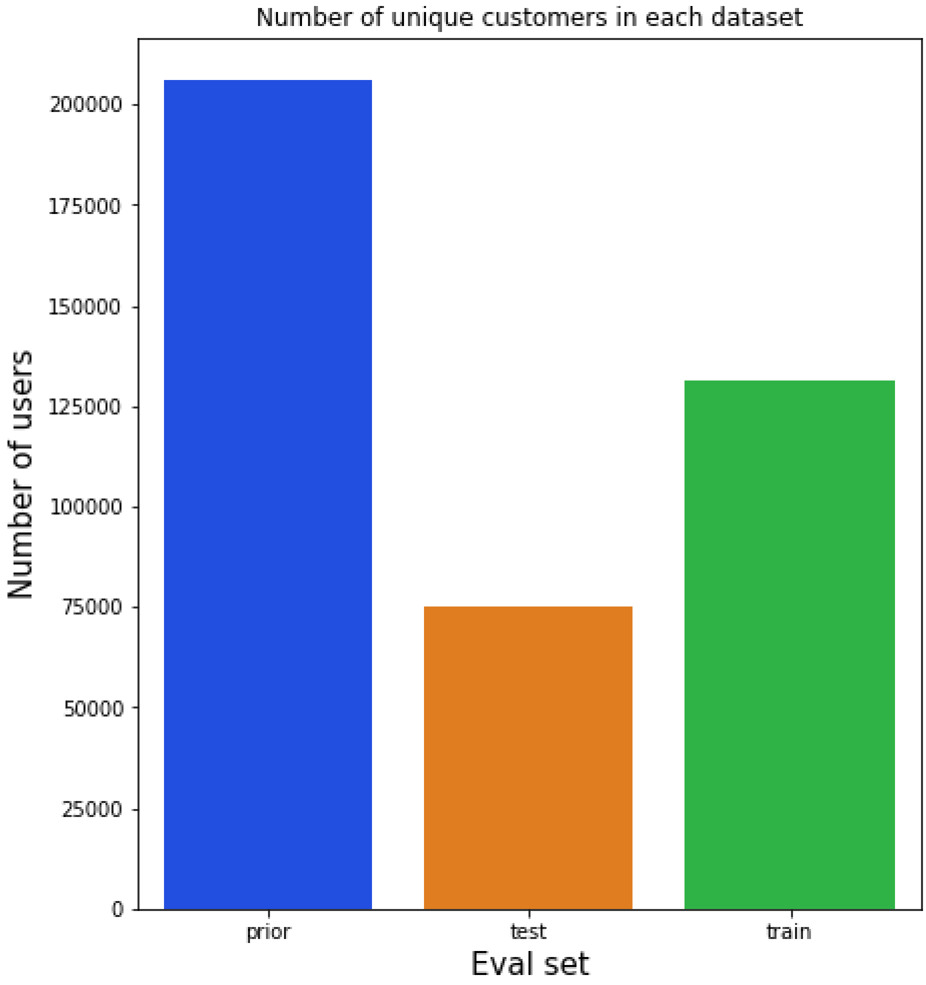
```
ordersDF.order_hour_of_day.unique()
n, bins, patches = plt.hist(ordersDF.order_hour_of_day, 47, facecolor="green", alpha=.75, a
plt.xlabel("Hour of Day")
plt.ylabel("Orders Count")
plt.title("When do people buy in a Day?")
plt.show()
```



In [13]:

```
grouped = ordersDF.groupby("eval_set")["user_id"].apply(lambda x: len(x.unique()))

plt.figure(figsize=(7,8))
sns.barplot(grouped.index, grouped.values, palette='bright')
plt.ylabel('Number of users', fontsize=15)
plt.xlabel('Eval set', fontsize=15)
plt.title("Number of unique customers in each dataset")
plt.show()
```



In [14]:

```
departmentsDF = pd.read_csv("C:/Users/Harsh Patel/Desktop/Big Data Mini Project/departments
aislesDF = pd.read_csv("C:/Users/Harsh Patel/Desktop/Big Data Mini Project/aisles.csv")
```

In [15]:

```
departmentsDF.head(20)
```

...

In [16]:

```
aislesDF.head(7)
```

Out[16]:

| | aisle_id | aisle |
|---|---|---|
| **0** | 1 | prepared soups salads |
| **1** | 2 | specialty cheeses |
| **2** | 3 | energy granola bars |
| **3** | 4 | instant foods |
| **4** | 5 | marinades meat preparation |
| **5** | 6 | other |
| **6** | 7 | packaged meat |

In [17]:

```
#Merge in single dataframe

items  = pd.merge(left =pd.merge(left=productsDF, right=departmentsDF, how='left'), right=a
items.head()
```

Out[17]:

| | product_id | product_name | aisle_id | department_id | department | aisle |
|---|---|---|---|---|---|---|
| **0** | 1 | Chocolate Sandwich Cookies | 61 | 19 | snacks | cookies cakes |
| **1** | 2 | All-Seasons Salt | 104 | 13 | pantry | spices seasonings |
| **2** | 3 | Robust Golden Unsweetened Oolong Tea | 94 | 7 | beverages | tea |
| **3** | 4 | Smart Ones Classic Favorites Mini Rigatoni Wit... | 38 | 1 | frozen | frozen meals |
| **4** | 5 | Green Chile Anytime Sauce | 5 | 13 | pantry | marinades meat preparation |

In [36]:

```
#Most important departments

grouped = items.groupby("department")["product_id"].aggregate({'Total_products': 'count'}).
grouped['Ratio'] = grouped["Total_products"].apply(lambda x: x /grouped['Total_products'].s
grouped.sort_values(by='Total_products', ascending=False, inplace=True)
grouped
```

```
C:\Anaconda\lib\site-packages\ipykernel_launcher.py:3: FutureWarning: using
a dict on a Series for aggregation
is deprecated and will be removed in a future version
  This is separate from the ipykernel package so we can avoid doing imports
until
```

Out[36]:

| | department | Total_products | Ratio |
|---|---|---|---|
| **17** | personal care | 6563 | 0.132084 |
| **20** | snacks | 6264 | 0.126067 |
| **16** | pantry | 5371 | 0.108095 |
| **3** | beverages | 4365 | 0.087848 |
| **10** | frozen | 4007 | 0.080643 |
| **7** | dairy eggs | 3449 | 0.069413 |
| **11** | household | 3085 | 0.062087 |
| **6** | canned goods | 2092 | 0.042103 |
| **9** | dry goods pasta | 1858 | 0.037393 |
| **19** | produce | 1684 | 0.033891 |
| **2** | bakery | 1516 | 0.030510 |
| **8** | deli | 1322 | 0.026606 |
| **14** | missing | 1258 | 0.025318 |
| **12** | international | 1139 | 0.022923 |
| **4** | breakfast | 1115 | 0.022440 |
| **1** | babies | 1081 | 0.021756 |
| **0** | alcohol | 1054 | 0.021212 |
| **18** | pets | 972 | 0.019562 |
| **13** | meat seafood | 907 | 0.018254 |
| **15** | other | 548 | 0.011029 |
| **5** | bulk | 38 | 0.000765 |

In [ ]:

# Recommendation Model

In [19]:

```
#Merging Datasets

opt = order_products_train.merge(productsDF,how='left', on='product_id')
opt = opt.merge(departmentsDF,how='left', on='department_id')
opt = opt.merge(aislesDF,how='left', on='aisle_id')
```

In [20]:

```
#Filtering of Data by reorders
reorders = opt[opt['reordered'] == 1]
reorders['product_id'] = reorders['product_id'].astype('int64')
```

```
C:\Anaconda\lib\site-packages\ipykernel_launcher.py:3: SettingWithCopyWarnin
g:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/s
table/indexing.html#indexing-view-versus-copy (http://pandas.pydata.org/pand
as-docs/stable/indexing.html#indexing-view-versus-copy)
  This is separate from the ipykernel package so we can avoid doing imports
 until
```

In [21]:

```
# get list of high volume products
hivol = reorders.copy()['product_id'].value_counts().sort_values(ascending=False)\
    [reorders.copy()['product_id'].value_counts().sort_values(ascending=False) > 1].index.t
```

In [22]:

```
# mask the reorders dataframe to only incluse dem hi vol prods
reorders = reorders[reorders['product_id'].isin(hivol)]
```

In [23]:

```
#filters the High demand items greater than.
reorders['hi_dem'] = (reorders.copy()['product_id'].value_counts().sort_values(ascending=Fa
```

In [24]:

```
hidem_ord = reorders[reorders['hi_dem'] == True]
```

In [25]:

```
user_orders = reorders.merge(ordersDF)
```

In [26]:

```
#reorders['hi_dem'] =
user_orders['hi_dem'] = (user_orders.copy()['product_id'].value_counts().sort_values(ascenc
```

In [27]:

```
hidem_ord = user_orders[user_orders['hi_dem'] == True]
```

# Setup of the model - compare users to another users

In [28]:

```
#return the total items
users = hidem_ord.groupby(['user_id','product_name']).size().sort_values(ascending=False).u
```

In [29]:

```
#creates a similiarity by users.
users_sim = pd.DataFrame(cosine_similarity(users),index=users.index,columns=users.index)
```

In [30]:

```
def next_prod(df,num_col):
    return df[df.columns[num_col]].drop(df.columns[num_col]).sort_values(ascending=False).h
```

In [31]:

```
#returns similar users to this one.
pd.DataFrame(next_prod(users_sim,56)).T
```

Out[31]:

| user_id | 43254 | 48962 | 10453 |
|---|---|---|---|
| **1711** | 0.5 | 0.353553 | 0.188982 |

# Recommendation for Products using userid

In [32]:

```
#return the total items in the basket from the aisles
products = hidem_ord.groupby(['product_name','user_id']).size().sort_values(ascending=False
```

In [33]:

```
#creates a similiarity by users.
products_sim = pd.DataFrame(cosine_similarity(products),index=products.index,columns=produc
```

In [34]:

```
#gives a recommendation for the last product added to shopping cart
pd.DataFrame(next_prod(products_sim,11)).T
```

Out[34]:

| product_name | Organic Heirloom Tomatoes | Organic Spinach Bunch | Salted Butter |
|---|---|---|---|
| **1% Low Fat Milk** | 0.57735 | 0.160128 | 0.144338 |

In [35]:

```
#Product Recommender by Order id

baskets = hidem_ord.groupby(['product_name','order_id']).size().sort_values(ascending=False
basket_sim = pd.DataFrame(cosine_similarity(baskets),columns=baskets.index,index=baskets.in
basket_sim['Green Peas'].sort_values(ascending=False).head(10)[1:]
```

Out[35]:

```
product_name
Small Compostable Waste Bag                        0.377964
Coho Salmon                                        0.377964
Unsweetened Vanilla                                0.377964
Gluten Free Broccoli & Cheese Baked Nuggets        0.377964
Rosemary Mini Croccantini                          0.267261
Organic Hearty Tomato Bisque                       0.267261
Veri Veri Teriyaki Marinade & Sauce Less Sodium    0.267261
Baby Food Meals                                    0.218218
Almond Milk Ricotta                                0.218218
Name: Green Peas, dtype: float64
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: