databricksShopper Behavior Exploration and Market Basket Analysis using Apache Spark

Market Basket Analysis is a technique used by large retailers to discover associations between their items. It works by looking for combinations of items that are bought together frequently, providing information to understand the purchase behavior. Association Rules Mining is one of the very important concepts of machine learning being used in Market Basket Analysis.

In this project, I will perform Shopper Behavior Exploration on a real Instacart Dataset of 3MM+ records and implement market basket analysis using Apache Spark MLlib FP-growth algorithm on Databricks.

Verifying the files we have in the databricks file system

```
%fs ls /FileStore/tables
```

path	name
dbfs:/FileStore/tables/aisles.csv	aisles.csv
dbfs:/FileStore/tables/departments.csv	departments.csv
dbfs:/FileStore/tables/order_products__prior.csv	order_products__prior.csv
dbfs:/FileStore/tables/order_products__train.csv	order_products__train.csv
dbfs:/FileStore/tables/orders.csv	orders.csv
dbfs:/FileStore/tables/products.csv	products.csv
dbfs:/FileStore/tables/sample_submission.csv	sample_submission.csv

Importing all the available files into the spark dataframe and creating temporary tables

```
aisles = spark.read.csv("/FileStore/tables/aisles.csv", header=True, inferSchema=True)
departments = spark.read.csv("/FileStore/tables/departments.csv", header=True, inferSchema=True)
order_products_prior = spark.read.csv("/FileStore/tables/order_products__prior.csv", header=True, inferSchema=True)
order_products_train = spark.read.csv("/FileStore/tables/order_products__train.csv", header=True, inferSchema=True)
orders = spark.read.csv("/FileStore/tables/orders.csv", header=True, inferSchema=True)
products = spark.read.csv("/FileStore/tables/products.csv", header=True, inferSchema=True)

aisles.createOrReplaceTempView("aisles")
departments.createOrReplaceTempView("departments")
order_products_prior.createOrReplaceTempView("order_products_prior")
order_products_train.createOrReplaceTempView("order_products_train")
orders.createOrReplaceTempView("orders")
products.createOrReplaceTempView("products")
```

Let's take a look at the top 5 rows of each of the imported file.

Top 5 orders in the orders dataframe

```
orders.show(n=5)
```

```
+--------+-------+--------+------------+---------+-----------------+-----------------------+
|order_id|user_id|eval_set|order_number|order_dow|order_hour_of_day|days_since_prior_order|
+--------+-------+--------+------------+---------+-----------------+-----------------------+
| 2539329|      1|   prior|           1|        2|                8|                   null|
| 2398795|      1|   prior|           2|        3|                7|                   15.0|
|  473747|      1|   prior|           3|        3|               12|                   21.0|
| 2254736|      1|   prior|           4|        4|                7|                   29.0|
|  431534|      1|   prior|           5|        4|               15|                   28.0|
+--------+-------+--------+------------+---------+-----------------+-----------------------+
only showing top 5 rows
```

Top 5 orders in the products dataframe

```
products.show(n=5)
```

```
+----------+--------------------+--------+-------------+
|product_id|        product_name|aisle_id|department_id|
+----------+--------------------+--------+-------------+
|         1|Chocolate Sandwic...|      61|           19|
|         2|    All-Seasons Salt|     104|           13|
|         3|Robust Golden Uns...|      94|            7|
|         4|Smart Ones Classi...|      38|            1|
|         5|Green Chile Anyti...|       5|           13|
+----------+--------------------+--------+-------------+
only showing top 5 rows
```

Top 5 orders in the order_products_train dataframe

```
order_products_train.show(n=5)
```

```
+--------+----------+-----------------+---------+
|order_id|product_id|add_to_cart_order|reordered|
+--------+----------+-----------------+---------+
|       1|     49302|                1|        1|
|       1|     11109|                2|        1|
|       1|     10246|                3|        0|
|       1|     49683|                4|        0|
|       1|     43633|                5|        1|
+--------+----------+-----------------+---------+
only showing top 5 rows
```

Top 5 orders in the order_products_prior dataframe

```
order_products_prior.show(n=5)
```

```
+--------+----------+-----------------+---------+
|order_id|product_id|add_to_cart_order|reordered|
+--------+----------+-----------------+---------+
|       2|     33120|                1|        1|
|       2|     28985|                2|        1|
|       2|      9327|                3|        0|
|       2|     45918|                4|        1|
|       2|     30035|                5|        0|
+--------+----------+-----------------+---------+
only showing top 5 rows
```

Top 5 orders in the departments dataframe

```
departments.show(n=5)
```

```
+-------------+----------+
|department_id|department|
+-------------+----------+
|            1|    frozen|
|            2|     other|
|            3|    bakery|
|            4|   produce|
|            5|   alcohol|
+-------------+----------+
only showing top 5 rows
```

Top 5 orders in the aisles dataframe

```
aisles.show(n=5)
```

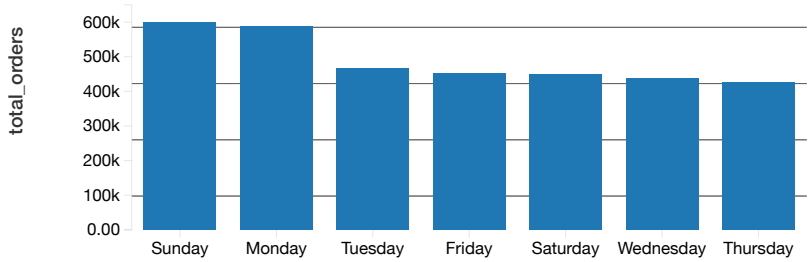
```
+--------+--------------------+
|aisle_id|               aisle|
+--------+--------------------+
|       1|prepared soups sa...|
|       2|   specialty cheeses|
|       3| energy granola bars|
|       4|       instant foods|
|       5|marinades meat pr...|
+--------+--------------------+
only showing top 5 rows
```

EXPLORATORY DATA ANALYSIS

On which day of the week customers purchase the most?

▶ Sunday and Monday have the most orders, while Thursday has the least orders in a week

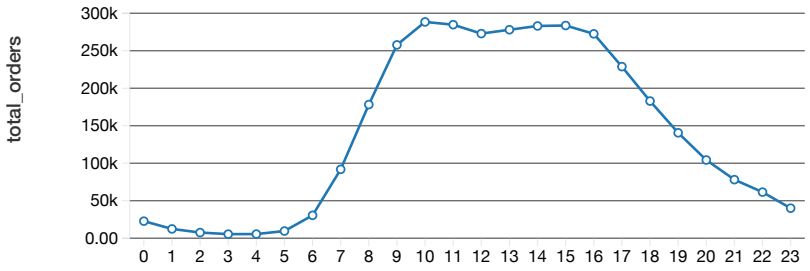
```
%sql
select count(order_id) as total_orders,
  (case
     when order_dow = '0' then 'Sunday'
     when order_dow = '1' then 'Monday'
     when order_dow = '2' then 'Tuesday'
     when order_dow = '3' then 'Wednesday'
     when order_dow = '4' then 'Thursday'
     when order_dow = '5' then 'Friday'
     when order_dow = '6' then 'Saturday'
   end) as day_of_week
  from orders
 group by order_dow
 order by total_orders desc
```



What time of day do customers purchase?

▶ Below line graph shows that the customers are more likely to place an order between 9 am to 6 pm

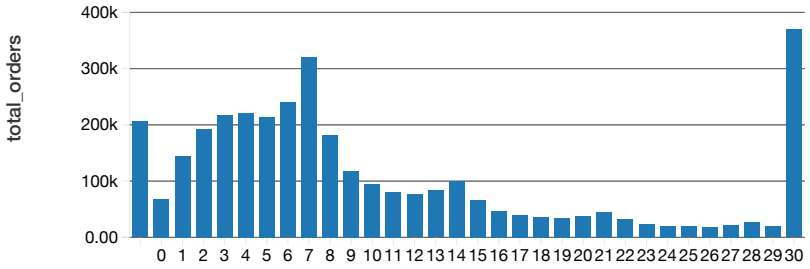
```
%sql
select count(order_id) as total_orders, order_hour_of_day as hour
from orders
group by order_hour_of_day
order by order_hour_of_day
```



How often do customers place orders?

▶ It appears that most of the customers order once a week since the majority of records are concentrated between 0 to 7 days ▶ Also, a large number of customer place their order 30 days or later days since because 'days_since_prior' column is capped at 30

```
%sql
select days_since_prior_order,count(order_id) as total_orders
from orders
group by days_since_prior_order
order by days_since_prior_order
```



Let's create a Master Dataset by merging together products, departments, order_products_train, and order_products_prior datasets together and run the query on top of that.

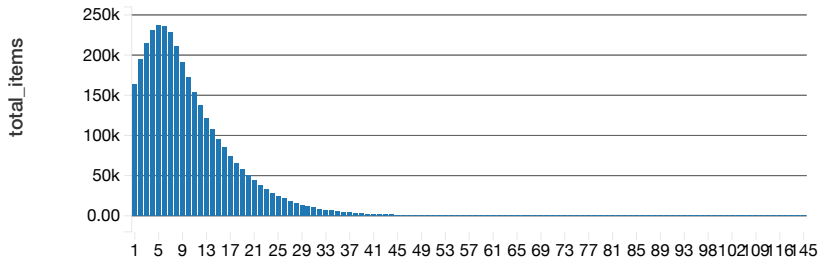
```
%sql
create table master_table as
(select op.*,p.product_name,p.aisle_id,p.department_id,d.department from
 (select * from order_products_train
 union
 select * from order_products_prior) as op
 inner join products as p
 on op.product_id = p.product_id
 inner join departments as d
 on p.department_id = d.department_id)
```

```
  Error in SQL statement: AnalysisException: Table default.master_table already exists. You need to drop it first.;
```

How many items do customers purchase in an order?

▶ The below bar chart depicts that the most common number of items purchased in order by customers is 4 ▶ Majority of customers prefer to purchase between 1 to 15 items per order

```
%sql
select order_id,count(product_id) as total_items
from master_table
group by order_id
```



Aggregated (by count) in the backend.

Which are the top departments from which orders are placed?

▶ If we take a look at top 10 departments from which most items are purchased, we would infer that almost 50% of the items purchased belong from just 2 departments which are 'produce' and 'dairy eggs'

```
%sql
select department, count(*) as orders_count from master_table
group by department
order by orders_count desc
limit 10
```

Which are the most purchased items?

▶ These are the top 8 items bought by Instacart customers in their orders. Banana seems to be most bought commonly bought item in baskets followed by strawberries, baby spinach, avocado, etc.

```
%sql
select product_name, count(*) as orders_count from master_table
group by product_name
order by orders_count desc
limit 200
```

product_name
Banana
Bag of Organic Bananas
Organic Strawberries
Organic Baby Spinach
Organic Hass Avocado
Organic Avocado

Large Lemon

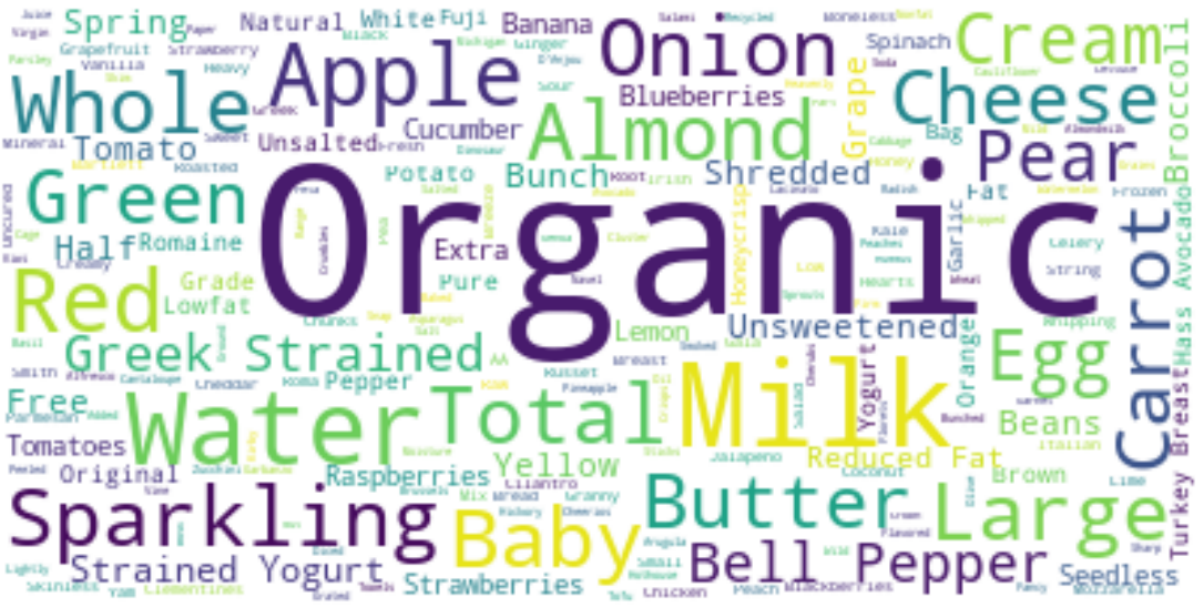
Let's also make a word cloud of the top 200 items bought by Instacart customers.

```
# !pip install wordcloud
from wordcloud import WordCloud
import matplotlib.pyplot as plt


df = sqlContext.sql("SELECT product_name FROM (select product_name, count(*) as orders_count from master_table group by
product_name order by orders_count desc limit 200)")
df2 = df.rdd.flatMap(lambda x: x).collect()
fullStr = ' '.join(df2)


wordcloud = WordCloud(background_color="white").generate(fullStr)

# Display the generated image:
plt.figure(figsize=(14, 10))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.show()
display()
```



From the word cloud, it appears that Americans buy organic food and veggies a lot as words like Organic, Milk, Water, Apple, Sparling, Egg, Green, Cheese, etc. are getting highlighted the most.

FP-Growth Algorithm

Organizing the data by the shopping basket:

For implementing FP-growth, first, we would be creating baskets of each order in our dataset. We would do so by creating a baskets data frame having 2 columns: first, the order_id and second, the list of items bought in that particular order.

```
# Organize the data by shopping basket
from pyspark.sql.functions import collect_set, col, count
rawData = spark.sql("select p.product_name, o.order_id from products p inner join order_products_train o where o.product_id =
p.product_id")
baskets = rawData.groupBy('order_id').agg(collect_set('product_name').alias('items'))
baskets.createOrReplaceTempView('baskets')
rawData.show(5)
baskets.show(5)
display(baskets)
```

order_id	items
1342	["Raw Shrimp","Seedless Cucumbers","Versatile Stain Remover","Organic Strawberries","Organic Mandarins","Chicken Apple Sausage","Pink Lady Ap
1591	["Cracked Wheat","Strawberry Rhubarb Yoghurt","Organic Bunny Fruit Snacks Berry Patch","Goodness Grapeness Organic Juice Drink","Honey Graha Roasted Turkey Breast","Pure Vanilla Extract","Chewy 25% Low Sugar Chocolate Chip Granola","Banana","Original Turkey Burgers Smoke Flavor Added Oranges","Lower Sugar Instant Oatmeal Variety","Ultra Thin Sliced Provolone Cheese","Natural Vanilla Ice Cream","Cinnamon Multigrain Cereal","Garlic Grain Chips","Medium Scarlet Raspberries","Lemon Yogurt","Original Patties (100965) 12 Oz Breakfast","Nutty Bars","Strawberry Banana Smoothie","G Cookies","Buttermilk Waffles","Uncured Genoa Salami","Organic Greek Whole Milk Blended Vanilla Bean Yogurt"]
4519	["Beet Apple Carrot Lemon Ginger Organic Cold Pressed Juice Beverage"]
4935	["Vodka"]
6357	["Globe Eggplant","Panko Bread Crumbs","Fresh Mozzarella Ball","Grated Parmesan","Gala Apples","Italian Pasta Sauce Basilico Tomato, Basil & Garli
10362	["Organic Baby Spinach","Organic Spring Mix","Organic Leek","Slow Roasted Lightly Seasoned Chick'n","Organic Basil","Organic Shredded Mild Chec

Showing the first 1000 rows.

Above are the top 5 rows of the baskets data frame, to be fed into the FP-growth algorithm.

Implementation of FP-growth algorithm using Scala:

Here, we would be using spark.ml's FP-growth package for implementation.

```
%scala
import org.apache.spark.ml.fpm.FPGrowth

// Extract out the items
val baskets_ds = spark.sql("select items from baskets").as[Array[String]].toDF("items")

// Use FPGrowth
val fpgrowth = new FPGrowth().setItemsCol("items").setMinSupport(0.001).setMinConfidence(0)
val model = fpgrowth.fit(baskets_ds)

// Display frequent itemsets
val mostPopularItemInABasket = model.freqItemsets
mostPopularItemInABasket.createOrReplaceTempView("mostPopularItemInABasket")

// Display generated association rules.
val ifThen = model.associationRules
ifThen.createOrReplaceTempView("ifThen")

import org.apache.spark.ml.fpm.FPGrowth
baskets_ds: org.apache.spark.sql.DataFrame = [items: array<string>]
fpgrowth: org.apache.spark.ml.fpm.FPGrowth = fpgrowth_056f6cf2dad5
model: org.apache.spark.ml.fpm.FPGrowthModel = fpgrowth_056f6cf2dad5
mostPopularItemInABasket: org.apache.spark.sql.DataFrame = [items: array<string>, freq: bigint]
ifThen: org.apache.spark.sql.DataFrame = [antecedent: array<string>, consequent: array<string> ... 2 more fields]
```

Now, let us explore the most frequent basket of items (containing at least 2 items).

```
%sql
select items, freq from mostPopularItemInABasket where size(items) > 2 order by freq desc limit 20
```

items
["Organic Hass Avocado","Organic Strawberries","Bag of Organic Bananas"]
["Organic Raspberries","Organic Strawberries","Bag of Organic Bananas"]
["Organic Baby Spinach","Organic Strawberries","Bag of Organic Bananas"]

▸["Organic Raspberries","Organic Hass Avocado","Bag of Organic Bananas"]
▸["Organic Hass Avocado","Organic Baby Spinach","Bag of Organic Bananas"]
▸["Organic Avocado","Organic Baby Spinach","Banana"]
▸["Organic Avocado","Large Lemon","Banana"]
▸["Limes","Large Lemon","Banana"]

The most frequent basket of items comprises of organic avocado, organic strawberries, and organic bananas together.

A good way to think about association rules is that model determines that if you purchased something (i.e. the antecedent), then you will purchase this other thing (i.e. the consequent) with the following confidence.

```
%sql
select antecedent as `antecedent (if)`, consequent as `consequent (then)`, confidence from ifThen order by confidence desc
limit 20
```

antecedent (if)	consequent (then)
▸["Organic Raspberries","Organic Hass Avocado","Organic Strawberries"]	▸["Bag of Organic Bananas"]
▸["Organic Cucumber","Organic Hass Avocado","Organic Strawberries"]	▸["Bag of Organic Bananas"]
▸["Organic Kiwi","Organic Hass Avocado"]	▸["Bag of Organic Bananas"]
▸["Organic Navel Orange","Organic Raspberries"]	▸["Bag of Organic Bananas"]
▸["Yellow Onions","Strawberries"]	▸["Banana"]
▸["Organic Whole String Cheese","Organic Hass Avocado"]	▸["Bag of Organic Bananas"]
▸["Organic Navel Orange","Organic Hass Avocado"]	▸["Bag of Organic Bananas"]
▸["Organic Raspberries","Organic Hass Avocado"]	▸["Bag of Organic Bananas"]
▸["Organic D'Anjou Pears","Organic Hass Avocado"]	▸["Bag of Organic Bananas"]

If a customer has organic raspberries, organic avocados, and organic strawberries in its basket, then it may make sense to recommend organic bananas as well. Surprisingly, the top 10 purchase recommendations either organic bananas or bananas.

Implementation of FP-growth algorithm — Market basket analysis using PySpark:

```
from pyspark.ml.fpm import FPGrowth

fpGrowth = FPGrowth(itemsCol="items", minSupport=0.001, minConfidence=0)
model = fpGrowth.fit(baskets)

# Display frequent itemsets.
model.freqItemsets.show()

# Display generated association rules.
model.associationRules.show()

# transform examines the input items against all the association rules and summarize the consequents as prediction
model.transform(baskets).show()
```

```
+--------------------+-----+
|               items| freq|
+--------------------+-----+
|[Organic Tomato B...|  772|
|[Organic Tomato B...|  175|
|[Organic Tomato B...|  144|
|[Organic Tomato B...|  179|
|[Organic Spinach ...|  475|
|[Whole Milk Ricot...|  347|
| [Medium Salsa Roja]|  275|
|    [Ground Buffalo]|  231|
|       [Tonic Water]|  194|
|[Original Coconut...|  173|
|[Low-Fat Strawber...|  152|
```

```
|[Organic SprouTof...|  137|
|            [Banana]|18726|
|[Fruit Punch Spor...|  275|
|[Kitchen Cheese E...|  230|
|[Country White Br...|  194|
|[Soft & Smooth Wh...|  173|
|[Natural Liquid L...|  152|
```