

 databricks Market Basket Analysis using Instacart Online Grocery

(<http://databricks.com/Dataset>)

# Market Basket Analysis using Instacart Online Grocery Dataset

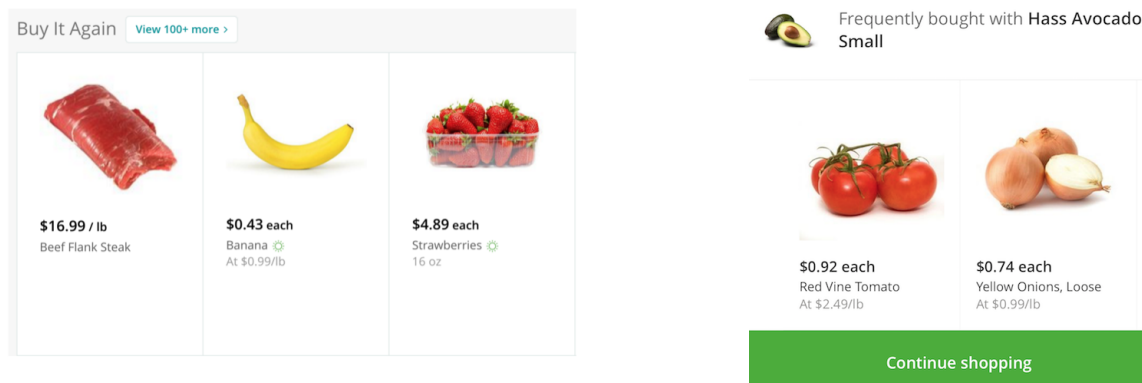
## Which products will an Instacart consumer purchase again?

To showcase the concept of market basket analysis with the Databricks Unified Analytics Platform (<https://databricks.com/product/unified-analytics-platform>), we will use the Instacart's 3 Million Instacart Orders, Open Sourced (<https://www.instacart.com/datasets/grocery-shopping-2017>) dataset.

“The Instacart Online Grocery Shopping Dataset 2017”, Accessed from <https://www.instacart.com/datasets/grocery-shopping-2017> (<https://www.instacart.com/datasets/grocery-shopping-2017>) on 01/17/2018. This anonymized dataset contains a sample of over 3 million grocery orders from more than 200,000 Instacart users. For each user, we provide between 4 and 100 of their orders, with the sequence of products purchased in each order. We also provide the week and hour of day the order was placed, and a relative measure of time between orders.

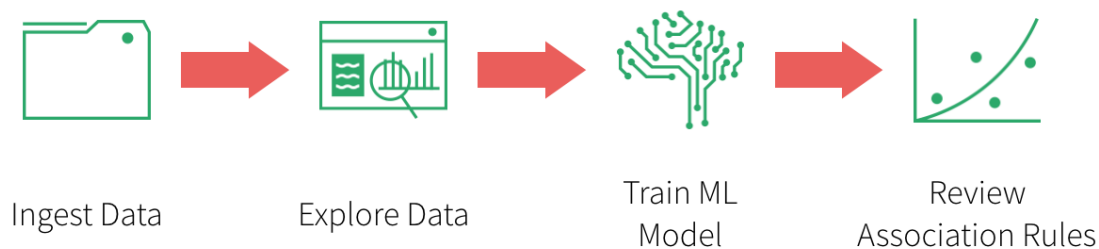
Whether you shop from meticulously planned grocery lists or let whimsy guide your grazing, our unique food rituals define who we are. Instacart's grocery ordering and delivery app aims to make it easy to fill your refrigerator and pantry with your personal favorites and staples when you need them. After selecting products through the Instacart app, personal shoppers review your order and do the in-store shopping and delivery for you.

This notebook will show how you can predict which items a shopper will purchase whether they buy it again or recommend to try for the first time.



Source: 3 Million Instacart Orders, Open Sourced (<https://tech.instacart.com/3-million-instacart-orders-open-sourced-d40d29ead6f2>)

## Data Engineering Pipeline



Data engineering pipelines are commonly comprised of these components:

- **Ingest Data:** Bringing in the data from your source systems; often involving ETL processes (though we will skip this step in this demo for brevity)
- **Explore Data:** Now that you have cleansed data, explore it so you can get some business insight
- **Train ML Model:** Execute FP-growth for frequent pattern mining
- **Review Association Rules:** Review the generated association rules

## Ingest Data

First, download the 3 Million Instacart Orders, Open Sourced (<https://www.instacart.com/datasets/grocery-shopping-2017>) and upload it to `dbfs` ; for more information, refer to Importing Data (<https://docs.databricks.com/user-guide/importing-data.html>).

The following `dbutils filesystem (fs)` query displays the six files:

- Orders : 3.4M rows, 206K users
- Products : 50K rows
- Aisles : 134 rows
- Departments : 21 rows
- order\_products\_\_SET : 30M+ rows where SET is defined as:
  - prior : 3.2M previous orders
  - train : 131K orders for your training dataset

Reference: The Instacart Online Grocery Shopping Dataset 2017 Data Descriptions (<https://gist.github.com/jeremystan/c3b39d947d9b88b3ccff3147dbcf6c6b>)


## Important

You will need to **edit** the locations (in the examples below, we're using `/mnt/bhavin/mba/instacart/csv` ) to where you had uploaded your data.

## Review Ingested Files

```
%fs ls /mnt/bhavin/mba/instacart/csv
```

| path   |
|--|
| dbfs:/mnt/bhavin/mba/instacart/csv/aisles.csv                |
| dbfs:/mnt/bhavin/mba/instacart/csv/departments.csv           |
| dbfs:/mnt/bhavin/mba/instacart/csv/order_products__prior.csv |
| dbfs:/mnt/bhavin/mba/instacart/csv/order_products__train.csv |
| dbfs:/mnt/bhavin/mba/instacart/csv/orders.csv                |
| dbfs:/mnt/bhavin/mba/instacart/csv/products.csv              |



## Review `orders.csv` file

```
%fs head dbfs:/mnt/bhavin/mba/instacart/csv/orders.csv
```

```
[Truncated to first 65536 bytes]
order_id,user_id,eval_set,order_number,order_dow,order_hour_of_day,days_since_
prior_order
2539329,1,prior,1,2,08,
2398795,1,prior,2,3,07,15.0
473747,1,prior,3,3,12,21.0
2254736,1,prior,4,4,07,29.0
431534,1,prior,5,4,15,28.0
3367565,1,prior,6,2,07,19.0
550135,1,prior,7,1,09,20.0
3108588,1,prior,8,1,14,14.0
2295261,1,prior,9,1,16,0.0
2550362,1,prior,10,4,08,30.0
1187899,1,train,11,4,08,14.0
2168274,2,prior,1,2,11,
1501582,2,prior,2,5,10,10.0
1901567,2,prior,3,1,10,3.0
738281,2,prior,4,2,10,8.0
1673511,2,prior,5,3,11,8.0
1199898,2,prior,6,2,09,13.0
3194192,2,prior,7,2,12,14.0
```

## Create DataFrames

```
# Import Data
aisles = spark.read.csv("/mnt/bhavin/mba/instacart/csv/aisles.csv",
header=True, inferSchema=True)
departments = spark.read.csv("/mnt/bhavin/mba/instacart/csv/departments.csv",
header=True, inferSchema=True)
order_products_prior =
spark.read.csv("/mnt/bhavin/mba/instacart/csv/order_products__prior.csv",
header=True, inferSchema=True)
order_products_train =
spark.read.csv("/mnt/bhavin/mba/instacart/csv/order_products__train.csv",
header=True, inferSchema=True)
orders = spark.read.csv("/mnt/bhavin/mba/instacart/csv/orders.csv",
header=True, inferSchema=True)
products = spark.read.csv("/mnt/bhavin/mba/instacart/csv/products.csv",
header=True, inferSchema=True)

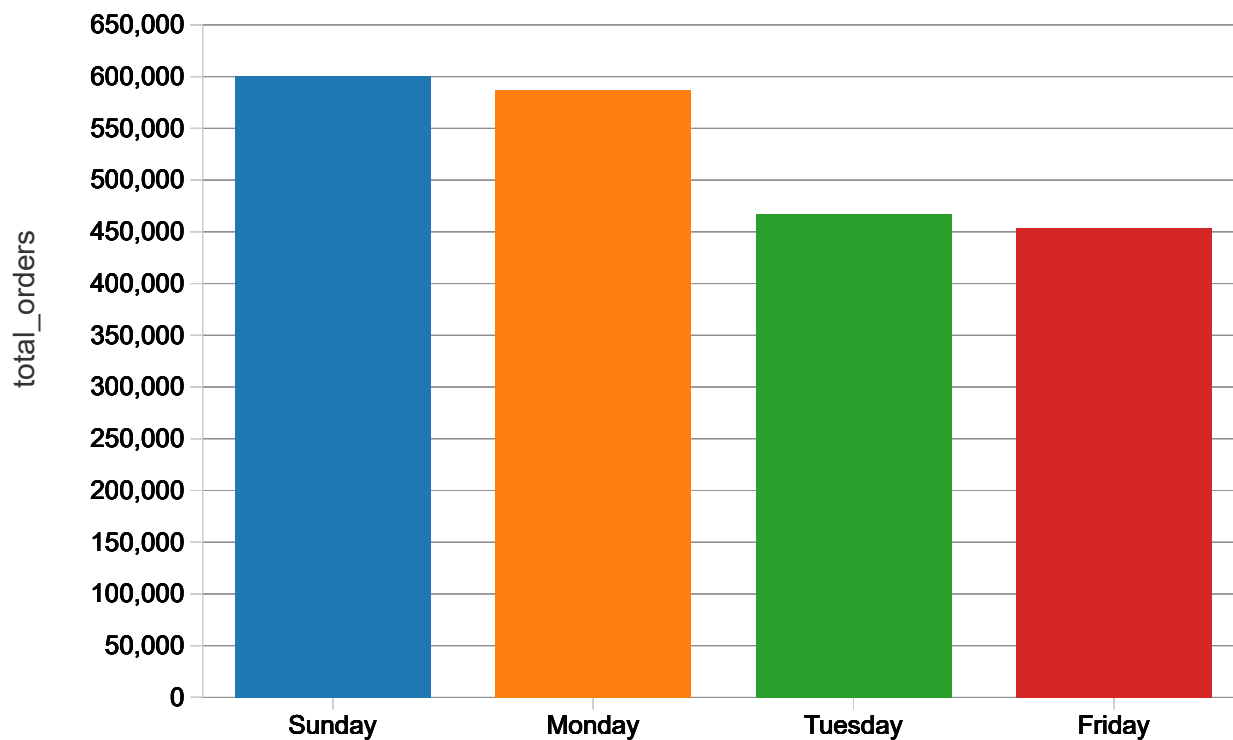
# Create Temporary Tables
aisles.createOrReplaceTempView("aisles")
departments.createOrReplaceTempView("departments")
order_products_prior.createOrReplaceTempView("order_products_prior")
order_products_train.createOrReplaceTempView("order_products_train")
orders.createOrReplaceTempView("orders")
products.createOrReplaceTempView("products")
```

# Exploratory Data Analysis

Explore your Instacart data using Spark SQL

## Busiest day of the 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
```



## Breakdown of Orders by Hour of the Day

```
%sql
```

```
select
```

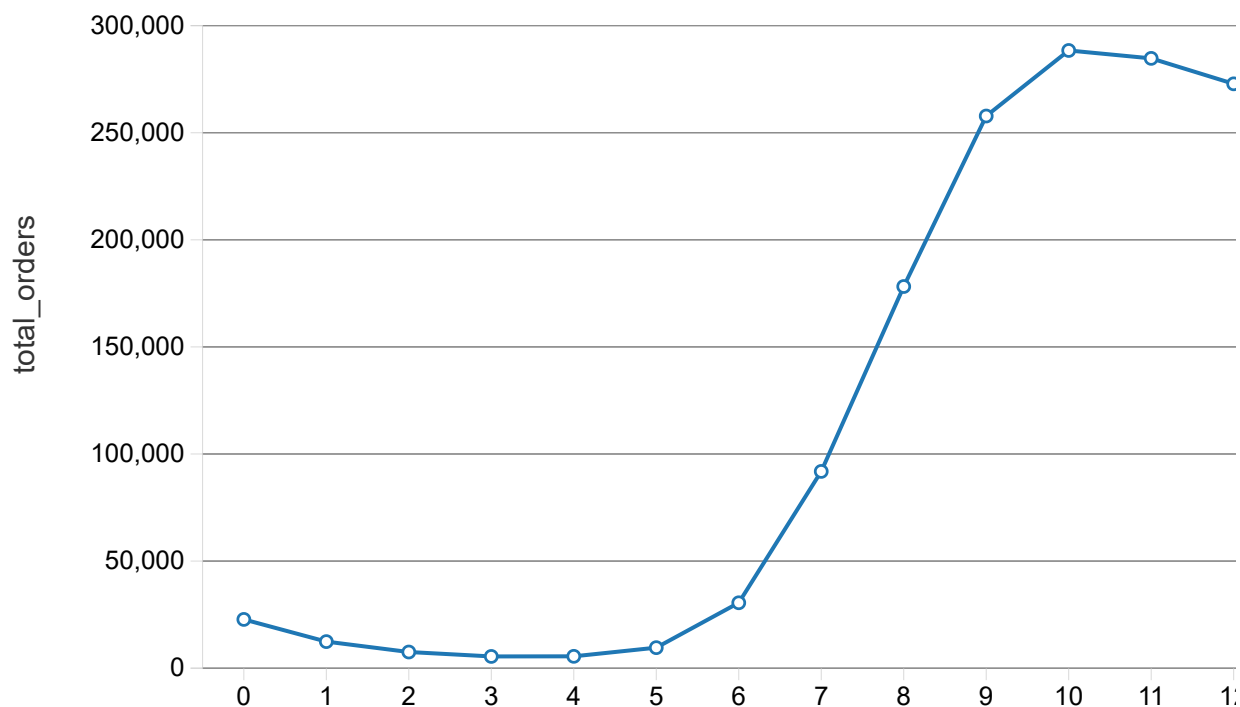
```
  count(order_id) as total_orders,
```

```
  order_hour_of_day as hour
```

```
from orders
```

```
group by order_hour_of_day
```

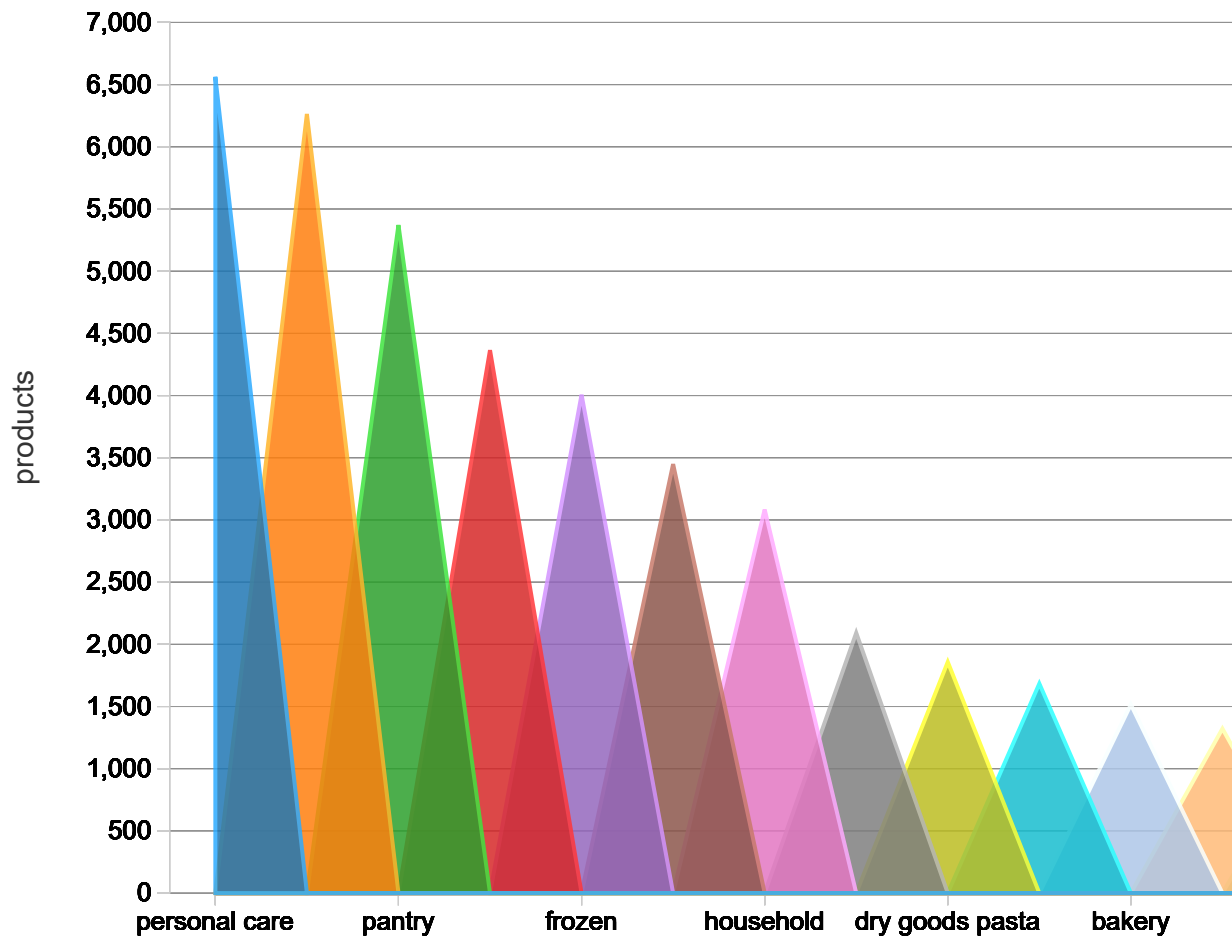
```
order by order_hour_of_day
```



## Max Products by Department

```
%sql
select countbydept.*
  from (
    -- from product table, let's count number of records per dept
    -- and then sort it by count (highest to lowest)
    select department_id, count(1) as counter
      from products
     group by department_id
     order by counter asc
  ) as maxcount
inner join (
  -- let's repeat the exercise, but this time let's join
  -- products and departments tables to get a full list of dept and
  -- prod count
  select
    d.department_id,
    d.department,
    count(1) as products
  from departments d
     inner join products p
       on p.department_id = d.department_id
  group by d.department_id, d.department
  order by products desc
) countbydept
-- combine the two queries's results by matching the product count
on countbydept.products = maxcount.counter
```





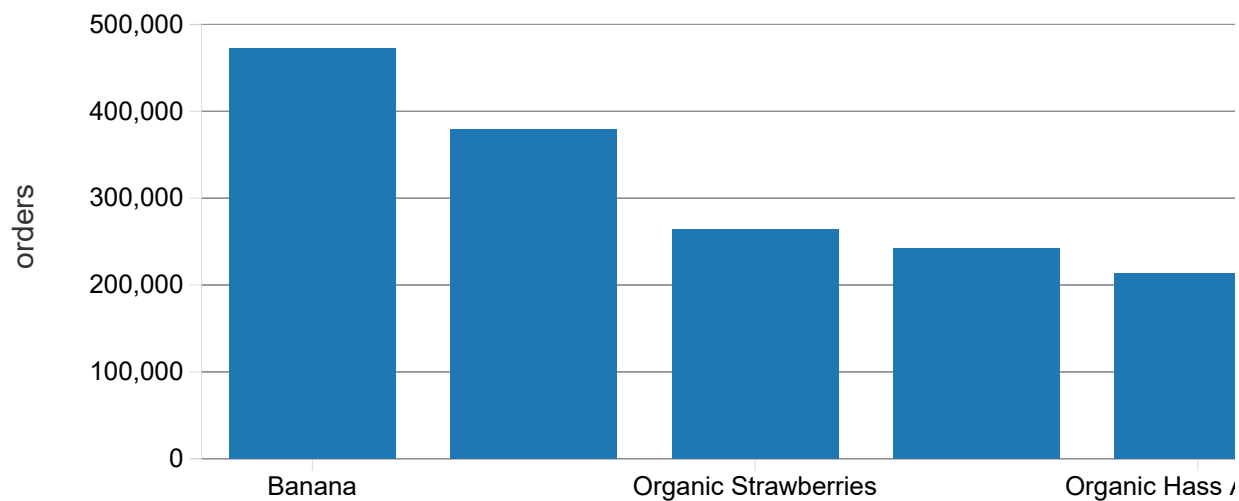
Only showing the first twenty series.



## Top 10 Popular Items

```
%sql
```

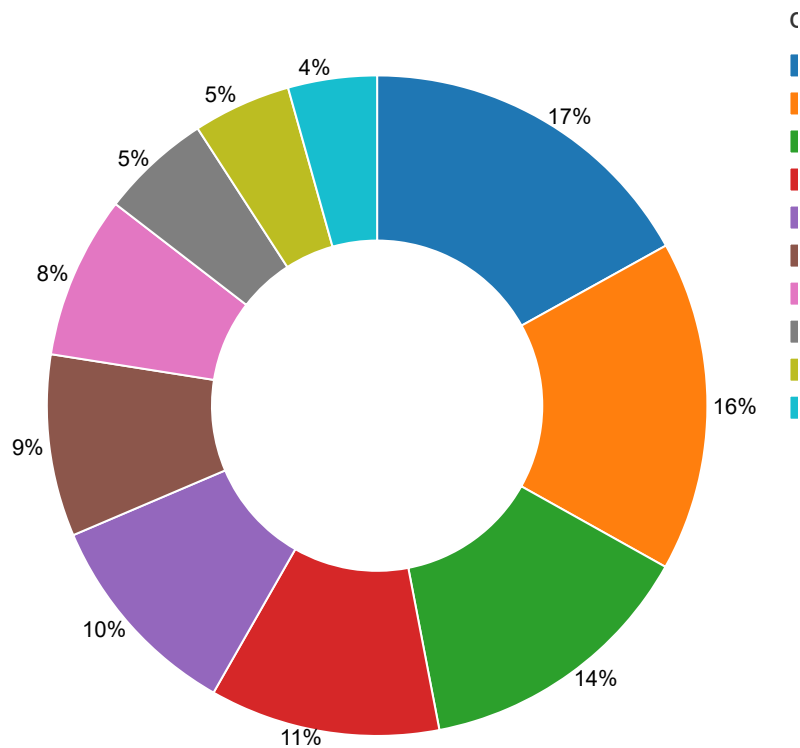
```
select count(opp.order_id) as orders, p.product_name as popular_product
  from order_products_prior opp, products p
 where p.product_id = opp.product_id
 group by popular_product
 order by orders desc
 limit 10
```



## Shelf Space by Department

%sql

```
select d.department, count(distinct p.product_id) as products
from products p
inner join departments d
on d.department_id = p.department_id
group by d.department
order by products desc
limit 10
```



## Organize and View Shopping Basket

### Organize Shopping Basket

```
# 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')
```

### View Shopping Basket

```
display(baskets)
```

| order_id | items   |
|----------|---|
| 1342     | ▶["Raw Shrimp","Seedless Cucumbers","Versatile Stain Remover","Organic Strawberries"] |

|       |   |
|-------|---|
| 1591  | ▶["Cracked Wheat","Strawberry Rhubarb Yoghurt","Organic Bunny Fruit Snacks Berry Pa<br>Vanilla Extract","Chewy 25% Low Sugar Chocolate Chip Granola","Banana","Original Tur<br>Sliced Provolone Cheese","Natural Vanilla Ice Cream","Cinnamon Multigrain Cereal","Ga<br>12 Oz Breakfast","Nutty Bars","Strawberry Banana Smoothie","Green Machine Juice Smc         |
| 4519  | ▶["Beet Apple Carrot Lemon Ginger Organic Cold Pressed Juice Beverage"]   |
| 4935  | ▶["Vodka"]  |
| 6357  | ▶["Globe Eggplant","Panko Bread Crumbs","Fresh Mozzarella Ball","Grated Parmesan","   |
| 10362 | ▶["Organic Baby Spinach","Organic Spring Mix","Organic Leek","Slow Roasted Lightly Se<br>Flour","Organic Gala Apples","Lemons","Limes","Pitted Dates","Jalapeno Peppers","Orig<br>Tomato","Organic Pinto Beans","Organic Three Grain Tempeh","Organic Garnet Sweet P<br>Avocados","Multigrain Tortilla Chips","The Ultimate Beefless Burger","Yellow Bell Pepper" ▼ |

Showing the first 1000 rows.

## Train ML Model

To understand the frequency of items are associated with each other (e.g. peanut butter and jelly), we will use association rule mining for market basket analysis. Spark MLlib (<http://spark.apache.org/docs/latest/mllib-guide.html>) implements two algorithms related to frequency pattern mining (FPM): FP-growth (<https://spark.apache.org/docs/latest/mllib-frequent-pattern-mining.html#fp-growth>) and PrefixSpan (<https://spark.apache.org/docs/latest/mllib-frequent-pattern-mining.html#prefixspan>). The distinction is that FP-growth does not use order information in the itemsets, if any, while PrefixSpan is designed for sequential pattern mining where the itemsets are ordered. We will use FP-growth as the order information is not important for this use case.

Note, we will be using the `Scala API` so we can configure `setMinConfidence` .

## Use FP-growth

```
%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)

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_1bacd1c33f93
model: org.apache.spark.ml.fpm.FPGrowthModel = fpgrowth_1bacd1c33f93
```

## Most Frequent Itemsets

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

mostPopularItemInABasket: org.apache.spark.sql.DataFrame = [items: array<string>, freq: bigint]

%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"]   |
| ▶ ["Organic Cucumber","Organic Strawberries","Bag of Organic Bananas"]     |
| ▶ ["Organic Baby Spinach","Organic Strawberries","Banana"]                 |

# Review Association Rules

In addition to `freqItemSets`, the `FP-growth` model also generates `association rules`. For example, if a shopper purchases *peanut butter*, what is the likelihood that they will also purchase *jelly*. For more information, a good reference is Susan Li's A Gentle Introduction on Market Basket Analysis—Association Rules (<https://towardsdatascience.com/a-gentle-introduction-on-market-basket-analysis-association-rules-fa4b986a40ce>)

## View Generated Association Rules

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

```
ifThen: org.apache.spark.sql.DataFrame = [antecedent: array<string>, consequent: array<string> ... 1 more field]
```

| antecedent (if)  |
|--|
| ▶ ["Organic Raspberries","Organic Hass Avocado","Organic Strawberries"]  |
| ▶ ["Organic Kiwi","Organic Hass Avocado"]                                |
| ▶ ["Organic Hass Avocado","Organic Baby Spinach","Organic Strawberries"] |
| ▶ ["Organic Whole String Cheese","Organic Hass Avocado"]                 |
| ▶ ["Yellow Onions","Strawberries"]                                       |
| ▶ ["Organic Navel Orange","Organic Raspberries"]                         |
| ▶ ["Organic Navel Orange","Organic Hass Avocado"]                        |
| ▶ ["Organic Cucumber","Organic Hass Avocado","Organic Strawberries"]     |
| ▶ ["Organic Fuji Apple","Seedless Red Grapes"]                           |
| ▶ ["Organic Raspberries","Organic Hass Avocado"]                         |
| ▶ ["Organic Unsweetened Almond Milk","Organic Hass Avocado"]             |
| ▶ ["Organic Fuji Apple","Strawberries"]                                  |
| ▶ ["Organic D'Anjou Pears","Organic Hass Avocado"]                       |
| ▶ ["Organic Gala Apples","Organic Hass Avocado"]                         |

