Market Basket Analysis on 3 million orders from Instacart using Spark.

Saket Garodia

Feb 29 · 8 min read

*Using the **FP-growth** algorithm to analyze associations between products of a supermarket*

Supermarkets around the world are using data mining techniques to analyze the user buying pattern in order to make their business more efficient thereby increasing their business and fulfilling customer needs at the same time. One such application is determining which are the goods that a consumer tends to buy together and analyze the 'if-then' patterns to understand if a consumer buys A which are the B's to **recommend** to the consumer. This helps them in placing the right products at the right aisle thereby helping the consumer recall their need. This helps the consumers to stock up their refrigerators and homes according to their needs and decreases their purchasing time at the same time.

In this project, I will use **Instacart's real dataset from Kaggle** which contains data from 3 million grocery orders from 200,000 users. For each user, there are about 4–100 different orders based on how many times they have purchased from Instacart. Here's the link to the dataset:

https://www.kaggle.com/c/instacart-market-basket-analysis/data

I will be showing the implementation on spark owing to the fact that it runs very fast on **Databricks** as it uses distributed in-process computing and takes very less time even on such a large dataset. Luckily we have the **FP (Frequent-Pattern Mining) library** already on **spark** and thereby I will use that to understand the patterns. Let us start.

Note: Databricks provides users to use their community platform for free. So, feel free to replicate the code and see the magic of pattern mining.

Let us start with importing all the files into the spark data frame:

```
#checking the files we have in the databricks file system

%fs ls /FileStore/tables
```

path	name	size
dbfs:/FileStore/tables/__aisles-ca881.csv	__aisles-ca881.csv	226
dbfs:/FileStore/tables/__departments-c112e.csv	__departments-c112e.csv	226
dbfs:/FileStore/tables/__order_products__prior-5ba78.csv	__order_products__prior-5ba78.csv	226
dbfs:/FileStore/tables/__order_products__train-e8408.csv	__order_products__train-e8408.csv	226
dbfs:/FileStore/tables/__orders-074d8.csv	__orders-074d8.csv	226
dbfs:/FileStore/tables/__products-3d492.csv	__products-3d492.csv	226
dbfs:/FileStore/tables/aisles.csv	aisles.csv	2603
dbfs:/FileStore/tables/departments.csv	departments.csv	270
dbfs:/FileStore/tables/order_products__prior.csv	order_products__prior.csv	577550706
dbfs:/FileStore/tables/order_products__train.csv	order_products__train.csv	24680147
dbfs:/FileStore/tables/orders.csv	orders.csv	108968645
dbfs:/FileStore/tables/products.csv	products.csv	2166953

```
#enabling arrow just for a backup if pandas is needed at any point of time
import numpy as np
import pandas as pd

#Importing all the available files into the spark dataframe
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)

# Create Temporary Tables to work using sql like commands
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")
```

Here's the data dictionary for all the files:

1) orders (3.4m rows, 206k users):

order_id: order identifier

user_id: customer identifier

eval_set: which evaluation set this order belongs in (see SET described below)

order_number: the order sequence number for this user (1 = first, n = nth)

order_dow: the day of the week the order was placed on

order_hour_of_day: the hour of the day the order was placed on

days_since_prior: days since the last order, capped at 30 (with NAs for order_number = 1)

2) products (50k rows):

product_id: product identifier

product_name: name of the product

aisle_id: foreign key

department_id: foreign key

3) aisles (134 rows):

aisle_id: aisle identifier

aisle: the name of the aisle

4) departments (21 rows):

department_id: department identifier

department: the name of the department

5) order_products__SET (30m+ rows):

order_id: foreign key

product_id: foreign key

add_to_cart_order: order in which each product was added to cart

reordered: 1 if this product has been ordered by this user in the past, 0 otherwise where SET is one of the four following evaluation sets (eval_set in orders):

“prior”: orders prior to that user's most recent order (~3.2m orders)

“train”: training data supplied to participants (~131k orders)

“test”: test data reserved for machine learning competitions (~75k orders)

Let us now explore what all data we have in the different files using the .show function in spark and do some basic exploratory data analysis to become familiar with the data.

```
#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
```

```
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
```

```
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
```

```
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
```

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

```
+-------------+----------+
```

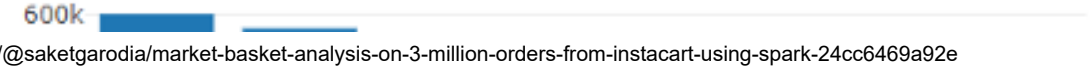
```
department_id|department|
+-------------+----------+
            1|    frozen|
            2|     other|
            3|    bakery|
            4|   produce|
            5|   alcohol|
+-------------+----------+
only showing top 5 rows
```

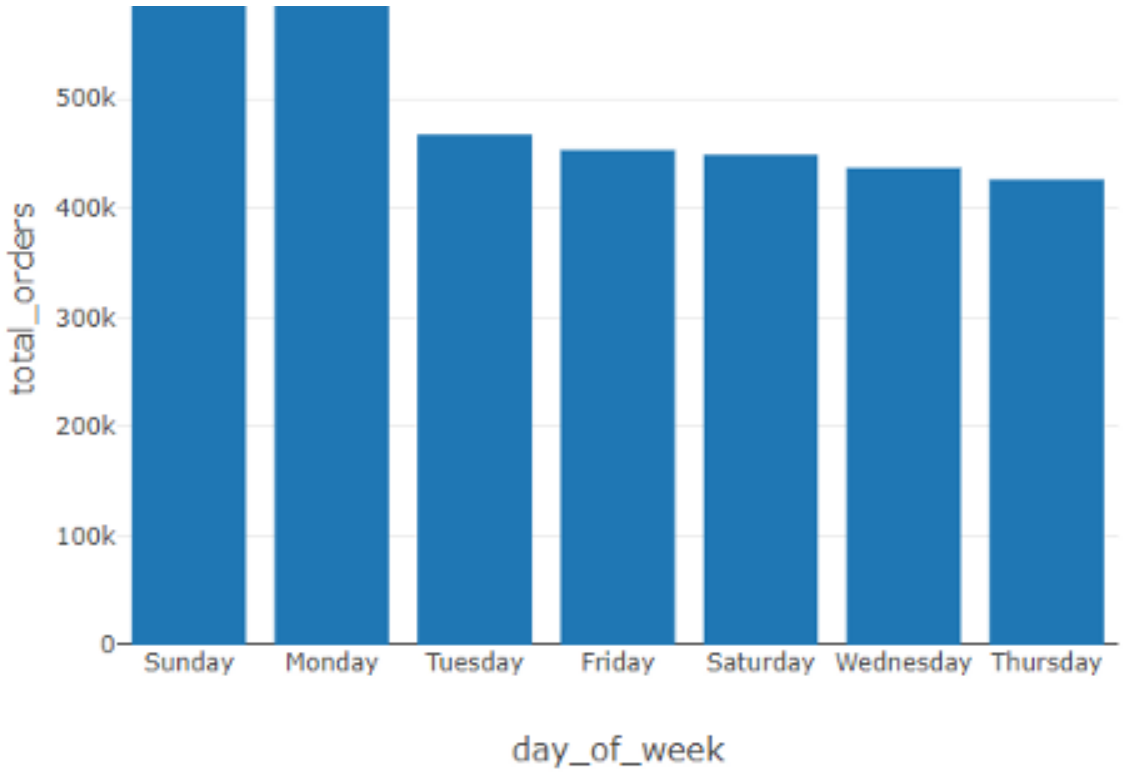
```
aisles.show(n=5)
```

```
+--------+--------------------+
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
```

Let us now see how the total number of orders differ for different days of the week.

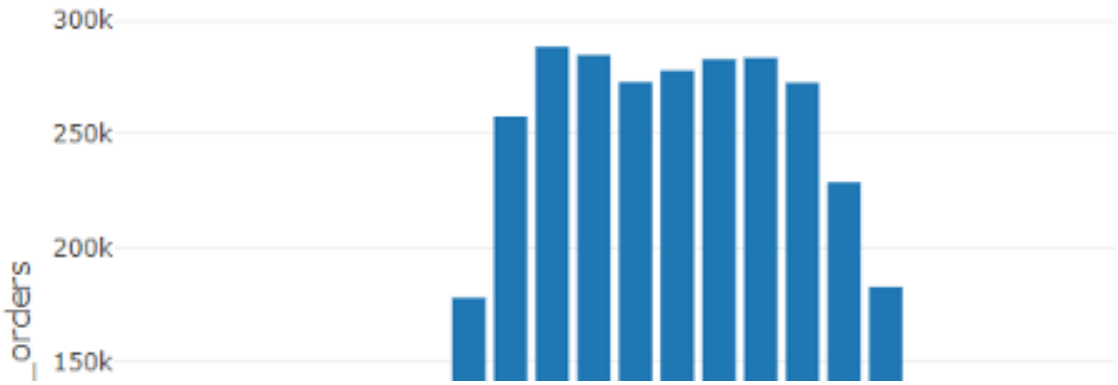
```
%sql
select
  count(order_id) as total_orders,
  (case
     when order_dow = '0' then 'Sunday'
     when order_dow = '1' then 'Monday'
     when order_dow = '2' then 'Tuesday'
     when order_dow = '3' then 'Wednesday'
     when order_dow = '4' then 'Thursday'
     when order_dow = '5' then 'Friday'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

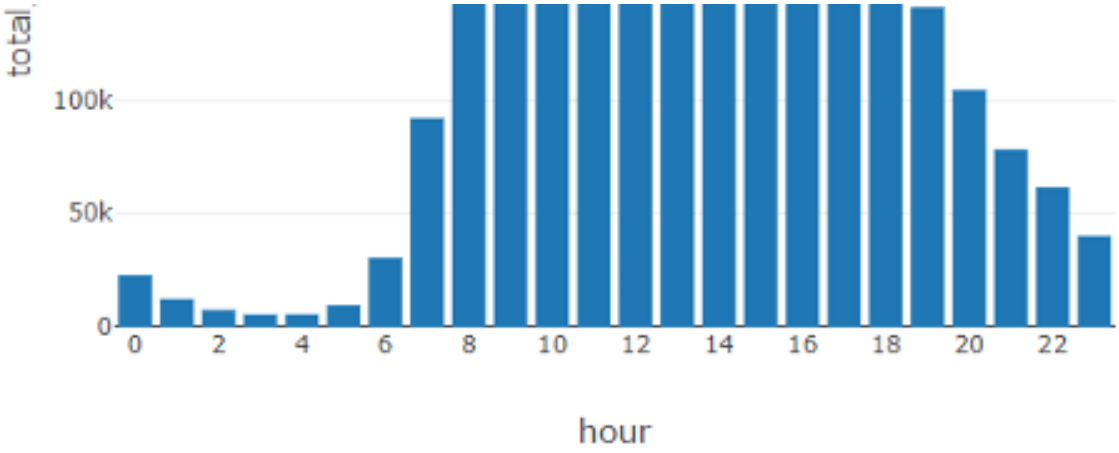




Most of the Instacart orders are placed on Sunday or Monday. Seems like people recall their needs just after their weekend fun ends. Jokes apart, let us also see the distribution across the time of a day to analyze the time when most of the users place an order from Instacart.

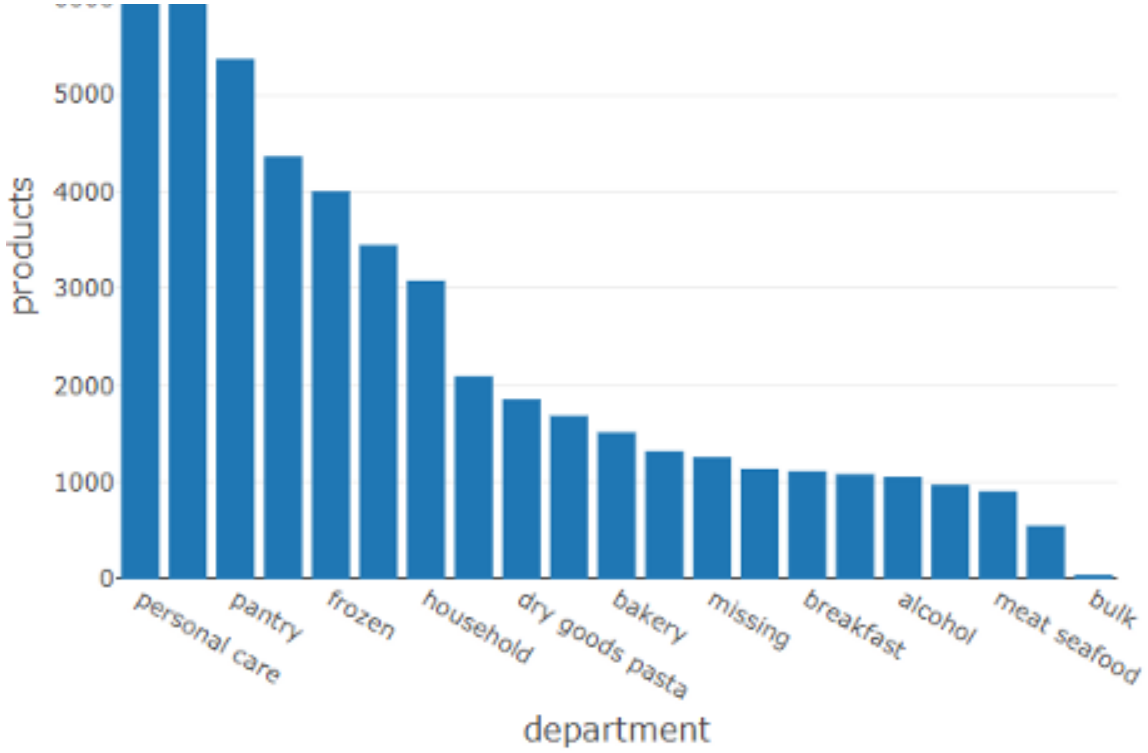
```
%sql
select
  count(order_id) as total_orders,
  order_hour_of_day as hour
  from orders
 group by order_hour_of_day
 order by order_hour_of_day
```





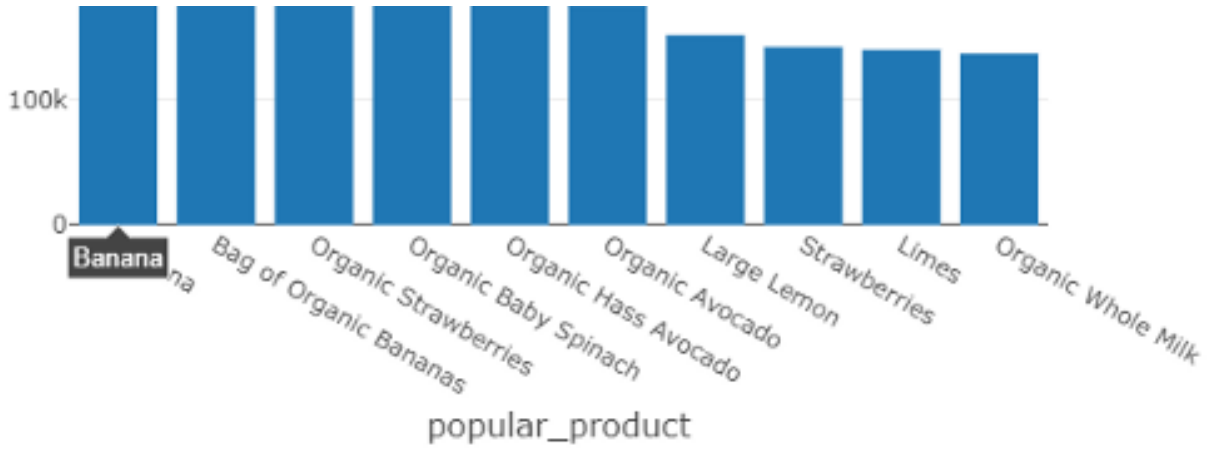
As we can see in the above bar plot, most of the orders are placed between 10 am and 4 pm. This was a bit surprising for me as I was expecting the peak to be outside office hours. Now, let us see which department is leading in terms of the number of products they have in the offerings.

```
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
```



The **personal care** department followed by the **pantry** and **frozen** leads in terms of the number of products they have. Now, let us visualize which products are the ones which are in the largest number of unique orders which means getting the most popular products.

```
%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
```



Bananas, Strawberries, and Spinach lead in terms of popularity. In fact, most of the top products seem healthy. When did America become so healthy?

Next, we have to prepare our data in the form that can be fed into the **pattern mining(FP growth) algorithm**. We need each row to have a basket of items that were ordered together. Let us create a baskets data frame for that before feeding into the algorithm.

```
# 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('it
ems'))
baskets.createOrReplaceTempView('baskets')

rawData.show(5)
baskets.show(5)

display(baskets)
```

order_id	items
1342	List(Raw Shrimp, Seedless Cucumbers, Versatile Stain Remover, Organic Strawberries, Organic Mandarins, Chicken Apple Sausage, Pink Lady Apples, Bag of Organic Bananas)
1591	List(Cracked Wheat, Strawberry Rhubarb Yoghurt, Organic Bunny Fruit Snacks Berry Patch, Goodness Grapeness Organic Juice Drink, Honey Graham Snacks, Spinach, Granny Smith Apples, Oven Roasted Turkey Breast, Pure Vanilla Extract, Chewy 25% Low Sugar Chocolate Chip Granola, Banana, Original Turkey Burgers Smoke Flavor Added, Twisted Tropical Tango Organic Juice Drink, Navel Oranges, Lower Sugar Instant Oatmeal Variety, Ultra Thin Sliced Provolone Cheese, Natural Vanilla Ice Cream, Cinnamon Multigrain Cereal, Garlic, Goldfish Pretzel Baked Snack Crackers, Original Whole Grain Chips, Medium Scarlet Raspberries, Lemon Yogurt, Original Patties (100965) 12 Oz

	Breakfast, Nutty Bars, Strawberry Banana Smoothie, Green Machine Juice Smoothie, Coconut Dreams Cookies, Buttermilk Waffles, Uncured Genoa Salami, Organic Greek Whole Milk Blended Vanilla Bean Yogurt)
4519	List(Beet Apple Carrot Lemon Ginger Organic Cold Pressed Juice Beverage)
4935	List(Vodka)
6357	List(Globe Eggplant, Panko Bread Crumbs, Fresh Mozzarella Ball, Grated Parmesan, Gala Apples, Italian Pasta Sauce Basilico Tomato, Basil & Garlic, Organic Basil, Banana, Provolone)
10362	List(Organic Baby Spinach, Organic Spring Mix, Organic Leek, Slow Roasted Lightly Seasoned Chick'n, Organic Basil, Organic Shredded Mild Cheddar, Bag of Organic Bananas, Sliced Baby Bella Mushrooms, Organic Tapioca Flour, Organic Gala Apples, Lemons, Limes, Pitted Dates, Jalapeno Peppers, Original Tofurky Deli Slices, Organic Red Bell Pepper, Organic Shredded Carrots, Roma Tomato, Crinkle Cut French Fries, Large Greenhouse Tomato, Organic Pinto Beans, Organic Three Grain Tempeh, Organic Garnet Sweet Potato (Yam), Organic Coconut Milk, Organic Extra Firm Tofu, Ground Sausage Style Veggie Protein, Extra Virgin Olive Oil, Hass Avocados, Multigrain Tortilla Chips, The Ultimate Beefless Burger, Yellow Bell Pepper, Coconut Flour, Light Brown Sugar, Organic Harissa Seasoning, Crushed Garlic, Organic Whole Cashews)

```
print((baskets.count(), len(baskets.columns)))
```

```
(131209, 2)
```

In total, we have about 1,31,209 basket of items. Now let us feed this data into the FPGrowth algorithm available in spark. Before doing that let us get some terms clear -

1) Support:

This measure gives an idea of how frequent an itemset is in all the transactions. Intuitively, for any basket A, support measures the % of transactions containing that basket as a subset.

2) Confidence:

This measure defines the likeliness of occurrence of consequent on the cart given that the cart already has the antecedents. Intuitively, let say there is a basket {a,b,c} having a support 's', then if we are analyzing ({a} implies {b,c}), confidence is the % of the transactions having {a,b,c} that contains {b,c}.

3) Lift:

Lift controls for the support (frequency) of consequent while calculating the conditional probability of occurrence of {Y} given {X}. Lift is the most important parameter that supermarkets use to place products. Think of it as the *lift* that {X} provides to our

confidence for having {Y} on the cart. To rephrase, lift is the rise in the probability of having {Y} on the cart with the knowledge of {X} being present over the probability of having {Y} on the cart without any knowledge about the presence of {X}.

Reference to understand these terms:

https://towardsdatascience.com/association-rules-2-aa9a77241654

Let us set the minimum support to 0.001, that means for our analysis, any basket that we will be analyzing should occur at least 0.001* 1,31,209 (131) times to be considered in our frequent pattern analysis.

```
%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)

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

Now, let us see the most frequent basket of items.

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

items	freq
List(Organic Hass Avocado, Organic Strawberries, Bag of Organic Bananas)	710
List(Organic Raspberries, Organic Strawberries, Bag of Organic Bananas)	649

List(Organic Baby Spinach, Organic Strawberries, Bag of Organic Bananas)	587
List(Organic Raspberries, Organic Hass Avocado, Bag of Organic Bananas)	531
List(Organic Hass Avocado, Organic Baby Spinach, Bag of Organic Bananas)	497
List(Organic Avocado, Organic Baby Spinach, Banana)	484
List(Organic Avocado, Large Lemon, Banana)	477
List(Limes, Large Lemon, Banana)	452
List(Organic Cucumber, Organic Strawberries, Bag of Organic Bananas)	424
List(Limes, Organic Avocado, Large Lemon)	389
List(Organic Raspberries, Organic Hass Avocado, Organic Strawberries)	381
List(Organic Avocado, Organic Strawberries, Banana)	379
List(Organic Baby Spinach, Organic Strawberries, Banana)	376
List(Organic Blueberries, Organic Strawberries, Bag of Organic Bananas)	374
List(Large Lemon, Organic Baby Spinach, Banana)	371
List(Organic Cucumber, Organic Hass Avocado, Bag of Organic Bananas)	366
List(Organic Lemon, Organic Hass Avocado, Bag of Organic Bananas)	353
List(Limes, Organic Avocado, Banana)	352

Now, let's use the association rules attribute of the FP-growth algorithm to analyze the if-then associations and see the confidence and lift values for different items. **For a rule to be helpful to Instacart, the lift value should be >1.**

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

%sql
select * from ifThen where lift > 1 order by lift desc
```

antecedent	consequent	confidence	lift
List(Strawberry Rhubarb Yoghurt)	List(Blueberry Yoghurt)	0.3096646942800789	80.29801358062228
List(Blueberry Yoghurt)	List(Strawberry Rhubarb Yoghurt)	0.3102766798418972	80.29801358062227
List(Icelandic Style Skyr Blueberry Non-fat Yogurt)	List(Nonfat Icelandic Style Strawberry Yogurt)	0.2170212765957447	78.66062066533443
List(Nonfat Icelandic Style Strawberry Yogurt)	List(Icelandic Style Skyr Blueberry Non-fat Yogurt)	0.42265193370165743	78.66062066533442

List(Icelandic Style Skyr Blueberry Non-fat Yogurt)	List(Non Fat Acai & Mixed Berries Yogurt)	0.2397163120567376	74.88794663964877
List(Non Fat Acai & Mixed Berries Yogurt)	List(Icelandic Style Skyr Blueberry Non-fat Yogurt)	0.4023809523809524	74.88794663964876
List(Blackberry Cucumber Sparkling Water)	List(Kiwi Sandia Sparkling Water)	0.25675675675675674	72.44902644580064
List(Kiwi Sandia Sparkling Water)	List(Blackberry Cucumber Sparkling Water)	0.2860215053763441	72.44902644580063
List(Icelandic Style Skyr Blueberry Non-fat Yogurt)	List(Non Fat Raspberry Yogurt)	0.3120567375886525	71.08446611505121
List(Non Fat Raspberry Yogurt)	List(Icelandic Style Skyr Blueberry Non-fat Yogurt)	0.3819444444444444	71.08446611505121
List(Lemon Sparkling Water)	List(Grapefruit Sparkling Water)	0.3130434782608696	65.19701863354038
List(Grapefruit Sparkling Water)	List(Lemon Sparkling Water)	0.22857142857142856	65.19701863354037
List(Total 2% Lowfat Greek Strained Yogurt	List(Total 2% Lowfat Greek		

As we can see in the above table which has the rules in decreasing value of the lift values, if someone buys [“Strawberry Rhubarb Yoghurt”] there is a very high chance of buying [“Blueberry Yoghurt”]

Displaying in the order of confidence leads to the following.

Note: Note quantifies the power of association that is unique because of the antecedent whereas confidence is just the probability of occurrence of the consequent when there is an antecedent.

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

antecedent (if)	consequent (then)	confidence
List(Organic Raspberries, Organic Hass Avocado, Organic Strawberries)	List(Bag of Organic Bananas)	0.5984251968503937
List(Organic Cucumber, Organic Hass Avocado, Organic Strawberries)	List(Bag of Organic Bananas)	0.546875
List(Organic Kiwi, Organic Hass Avocado)	List(Bag of Organic Bananas)	0.5459770114942529
List(Organic Navel Orange, Organic Raspberries)	List(Bag of Organic Bananas)	0.5412186379928315
List(Yellow Onions, Strawberries)	List(Banana)	0.5357142857142857
List(Organic Whole String Cheese, Organic Hass Avocado)	List(Bag of Organic Bananas)	0.5314685314685315
List(Organic Navel Orange, Organic Hass Avocado)	List(Bag of Organic Bananas)	0.5283018867924528
List(Organic Raspberries, Organic Hass Avocado)	List(Bag of Organic Bananas)	0.521099116781158
List(Organic D'Anjou Pears, Organic Hass Avocado)	List(Bag of Organic Bananas)	0.5170454545454546
List(Organic Unsweetened Almond Milk, Organic Hass Avocado)	List(Bag of Organic Bananas)	0.5141065830721003

List(Organic Broccoli, Organic Hass Avocado)	List(Bag of Organic Bananas)	0.5048231511254019
List(Organic Lemon, Organic Raspberries)	List(Bag of Organic Bananas)	0.4989106753812636
List(Organic Hass Avocado, Organic Baby Spinach, Organic Strawberries)	List(Bag of Organic Bananas)	0.49393939393939396
List(Organic Fuji Apple, Strawberries)	List(Banana)	0.4915254237288136

Reference: https://s3.us-east-2.amazonaws.com/databricks-dennylee/notebooks/Market+Basket+Analysis+using+Instacart+Online+Grocery+Dataset.html

Analytics Machine Learning Market Basket Analysis Spark Databricks