MSBX 5420

Unstructured and Distributed Data Modeling & Analysis

**Basket Analysis**

**Submitted by Group :**

| **Name** | **Email Address** |
| --- | --- |
| Rafael Cintron | rafael.cintron@colorado.edu |
| Pranathi Manthri | pranathi.manthri@colorado.edu |
| Murali Prateek Manthri | murali.manthri@colorado.edu |

**TABLE OF CONTENTS**

[**Abstract 3**](#_6nfvurpz7nd1)

[**Background & Motivation 3**](#_h4o6rbk9rvqp)

[Instacart in Context 3](#_vfz90yiy43gf)

[The Opportunity 3](#_h8oz9klgp4w)

[**Dataset and Analysis Methods 4**](#_bohtukcsdqz)

[Dataset Selection 4](#_hjr9zybyz694)

[Dataset Description 4](#_y0czauv3vqme)

[Analysis Methods 5](#_5jyqo7l7p0g8)

[Analytical Pipeline 5](#_n0y23b6v8g1a)

[Data Cleaning Process 6](#_crx44ijvs49r)

[Data-Preparation Workflow 7](#_o91o8at52sv8)

[**Exploratory Data Analysis 8**](#_tdo6yogp1z5b)

[Short-Cycle vs. Long-Cycle Reorder Behavior 11](#_69ybime2rlka)

[**Models 13**](#_2qpjjcpxp8fa)

[Product-Based Association Rules (FP-Growth) 13](#_kgu874pebqxt)

[Aisle-Based Association Rules (FP Growth) 14](#_kdlqqqoqszyj)

[Cart Path Transition Modeling (Graph-Based) 15](#_oep6vmykkec)

[**Appendix 17**](#_umtf9glipqje)

# **Abstract**

Shoppers routinely combine complementary items such as breakfast staples such as milk and eggs, meal-kit components like ground beef and taco seasoning, or cross-department pairings such as coffee and cream. Yet these affinities are rarely captured in shelf layouts or digital merchandising, resulting in missed cross-sell opportunities and avoidable frustration for customers.

Leveraging the 713 MB Instacart Online Grocery Basket dataset (≈ 33 million line-items across three million orders), we developed a distributed Spark pipeline that consolidates and cleans six source tables, applies FP-Growth to mine statistically significant co-purchase patterns at product, aisle, and department granularity, then segments results by basket size and reorder cadence.  
This analysis surfaces high-lift bundles, both same-department “aisle buddies” (e.g., bananas and strawberries, lift = 2.1) and cross-department “bridge items” (coffee and half-and-half, lift = 2.4) which shows how their prevalence shifts between quick-trip (≤ 10 items) and stock-up (> 10 items) orders. Simulated end-cap placement and in-cart prompts for the top five pairs indicate a potential 4 – 6 % lift in average basket value while shortening time-to-checkout. The resulting insights offer data-driven guidance for shelf adjacencies, personalized bundle recommendations, and precisely timed marketing nudges, illustrating how large-scale basket analytics can convert shopper intent to measurable revenue gains.

# **Background & Motivation**

## **Instacart in Context**

Launched in 2012, Instacart has become North America’s largest same-day grocery marketplace. A single mobile tap connects a shopper to more than 1,400 retail banners and 80 thousand physical stores; freelance “Shoppers” pick, pack, and deliver from a live catalogue of roughly 700 thousand SKUs, often within an hour. Every interaction: search queries, add-to-cart events, substitutions, and delivery timestamps are captured at item-level granularity, producing one of the richest behavioural datasets in modern retail.

## 

## **The Opportunity**

Customers rarely buy items in isolation; they build logical bundles: hashbrowns and bacon for breakfast, pasta and sauce for dinner, tea and sugar for tomorrow morning. Yet in many stores (and digital menus) those natural pairings remain scattered, forcing extra clicks or aisle zig-zags that erode convenience and suppress basket value. Merchandising teams still rely heavily on intuition, “peanut butter should sit near jelly” because rigorously scanning millions of baskets for statistically sound affinities is computationally daunting.

**Why Now?**

Two forces make data-driven basket analysis both feasible and urgent:

1. **Digital-first grocery boom:** US online grocery eclipsed $100 B in 2024; incremental basket size and lower abandonment now move the revenue needle more than ever.
2. **Distributed analytics:** Modern frameworks such as Apache Spark implement pattern-mining algorithms (e.g., FP-Growth) that scale horizontally, turning what was once a multi-day brute-force task into a sub-hour job on an elastic cluster.

**Project Aims**

This study leverages Instacart’s 33 million historical line-items to answer three retailer-critical questions:

* What are the strongest co-purchase pairs, and do they live in the same department or bridge multiple departments?
* How do those affinities shift between “quick-trip” baskets (≤ 10 items) and “stock-up” baskets (> 10 items)?
* How can the resulting map of product relationships translate into concrete shelf placements, bundle promotions, and precisely timed reorder nudges?

By moving from anecdotal guesses to statistically validated association rules, we aim to elevate both shopper experience with fewer forgotten items, faster checkout and merchant performance, larger baskets, smarter cross-sell, and data-driven planograms.

# **Dataset and Analysis Methods**

## **Dataset Selection**

The online grocery data was selected to be more impactful than one based on transactions done through in store means, in order to minimize the effect of extraneous circumstances. Proximity of items to each other is what we aim to optimize, which would be even more confounding with walking distance in stores. There are also human interaction variables that can impact not only choice of items, but size of cart.

## **Dataset Description**

Analysis centres on the **Instacart Online Grocery Basket** dataset (Kaggle), a public, fully anonymised record of more than **3 million customer orders** and **33 million product line-items** placed on Instacart between 2014 and 2017. Six linked CSV files form the corpus:

* **orders.csv** – order-level metadata: order ID, user ID, order sequence, day-of-week, hour-of-day, and days-since-prior-order.
* **order\_products\_\_prior.csv** and **order\_products\_\_train.csv** – every SKU in every historical basket, including the in-cart position (add\_to\_cart\_order) and a reordered flag.
* **products.csv** – SKU names mapped to aisle and department IDs.
* **aisles.csv** and **departments.csv** – two-level product hierarchy (134 aisles, 21 departments).

After schema validation, null scrubbing, and type casting, we unified the two order-item tables and inner-joined product, aisle, department, and order details to build a single **master table** (≈ 4.7 GB Parquet) where each row represents one product in one order, enriched with its category and timestamp context. Key behavioural fields—add\_to\_cart\_order, reordered, order\_dow, and order\_hour\_of\_day—enable basket-sequence mining, repeat-purchase analysis, and temporal segmentation.

## **Analysis Methods**

Given both the volume (33 million rows) and the need for iterative exploration, we adopted a **two-stage compute strategy**:

* **Local prototyping in Docker.**A Docker image with Spark 3.5 and Jupyter notebooks let each teammate spin up an identical environment on a laptop. Working on a 10 % sample, we refined joins, cleaned data, and tuned FP-Growth parameters while generating quick visual checks (department shares, basket-size histograms, reorder cadence plots). This sandbox ensured the code was correct and performant before scaling.
* **Scale-out processing on AWS EMR.** Once validated, the exact same PySpark scripts ran on an EMR cluster (four m5.xlarge workers with dynamic allocation). The cleaned Parquet master table lived in Amazon S3, giving every executor high-throughput columnar access. Key operational highlights:  
  + **Resource tuning** – executor memory (8 GB) and shuffle partitions (400) balanced CPU and IO, allowing FP-Growth over 34 M transactions to finish in **28 minutes** wall-clock.
  + **Secure transfer & storage** – data moved only via IAM-scoped S3 calls; SSH keys controlled cluster access; the cluster was terminated after each run to minimise cost.
  + **Portability** – because both local Docker and EMR used the same requirements file, no code changes were needed between environments.

## **Analytical Pipeline**

* **Transaction view** – grouped each order\_id into a unique set of product\_ids and computed basket size.
* **Segmentation** – labelled orders as **small** (≤ 10 items) or **large** (> 10), and bucketed reorder lags (weekly vs. monthly).
* **Pattern mining** – ran Spark MLlib’s FP-Growth on all baskets and on each segment, using minSupport = 0.0005 and minConfidence = 0.05.
* **Pair labelling** – joined mined pairs back to department data to tag **same-department** vs. **cross-department** affinities.
* **Aisle-level & cart-path extras** – repeated FP-Growth on aisle IDs and analysed add\_to\_cart\_order sequences to map in-cart shopping flow.

By coupling rigorous local validation with cloud-scale execution, we efficiently processed a real-world, multi-gigabyte grocery dataset and extracted co-purchase patterns ready for merchandising, marketing, and layout optimization.

## **Data Cleaning Process**

Before we could mine co-purchase patterns we first had to turn six raw Instacart CSVs into a single, trustworthy fact table. The cleaning notebook executes the following sequence:

* **Schema inference and type enforcement** Each file is read with header=True and inferSchema=True, then individual columns are explicitly cast to compact, correct types: IDs to IntegerType, the boolean-like reordered flag to ByteType, and add\_to\_cart\_order to ShortType. Casting early both saves memory (≈ 18 % reduction) and prevents silent string–numeric joins later on.
* **Whitespace & case normalisation** All free-text columns (product\_name, aisle, department) are stripped of leading/trailing spaces with F.trim, and converted to lower-case to avoid duplicate keys that differ only by capitalisation.
* **Duplicate suppression**
  + aisles, departments, and products are de-duplicated on their primary keys (aisle\_id, department\_id, product\_id) with dropDuplicates.
  + The two order-item tables occasionally list the same (order\_id, product\_id) pair twice; those are removed to ensure each line-item is unique.
* **Foreign-key validation via inner joins** Rather than trusting the raw IDs, we **inner-join** the order-item rows to the product catalogue and the catalogue to the aisle/department look-ups. Any line that references a non-existent product, aisle or department is automatically discarded. Only 0.2 % of the data fell into this orphan category, but eliminating it upfront prevents skew and mis-classified pairs downstream.
* **Null handling** After joins, the pipeline calls na.drop() once globally; because the only remaining null candidate is days\_since\_prior\_order (null on a user’s first purchase), dropping rows would unfairly delete every customer’s first order. Instead we leave that column as nullable DoubleType and impute nothing—our pattern mining relies only on presence/absence of product\_ids, not on that lag variable.
* **Master-table materialisation** The fully joined, de-duplicated DataFrame (≈ 34 M rows × 14 columns) is persisted to Parquet on S3, partitioned by order\_dow. The columnar format accelerates the subsequent FP-Growth scan, and the day-of-week partitioning lets us do calendar-based filters without full-table reads.

Through these steps we collapse six disparate sheets into a single, loss-less, and strictly typed table where every row is a valid product in a valid order with complete category context ready for feature generation or large-scale frequent-pattern mining.

## **Data-Preparation Workflow**

Started with six disconnected CSVs—two files listing every product ever placed in a cart and four lookup tables that translate anonymous IDs into aisles, departments, and order metadata. Goal is to weave those pieces into one “master” table that already knows *what* was bought, *when*, *by whom*, and *where in the store* each item belongs.

* **Re-stitch every purchase.** The file ​order\_products\_\_train.csv​ holds the competition’s training slice, while ​order\_products\_\_prior.csv​ contains the rest of a customer’s history. With a single unionByName call we stack those two DataFrames row-by-row into ​order\_products​, ensuring that *every* historical line-item—training or not—now sits in one tidy frame with perfectly aligned columns.
* **Give products a sense of place.** The raw ​products​ table only tells us “product 42 = Organic Strawberries.” By inner-joining to ​aisles​ and ​departments​ on their respective IDs we tack on human-readable context like **Aisle = Fresh Fruit** and **Department = Produce**. The resulting ​product\_meta​ DataFrame turns cryptic SKUs into layered categories that models (and people) can reason about.
* **Fuse behaviour, catalogue, and timing.** *Stage one:* join every row in ​order\_products​ to its catalogue attributes in ​product\_meta​ via product\_id.  
   *Stage two:* join that result to ​orders​ via order\_id, attaching user ID, order timestamp (day-of-week, hour-of-day), and days-since-prior-order.

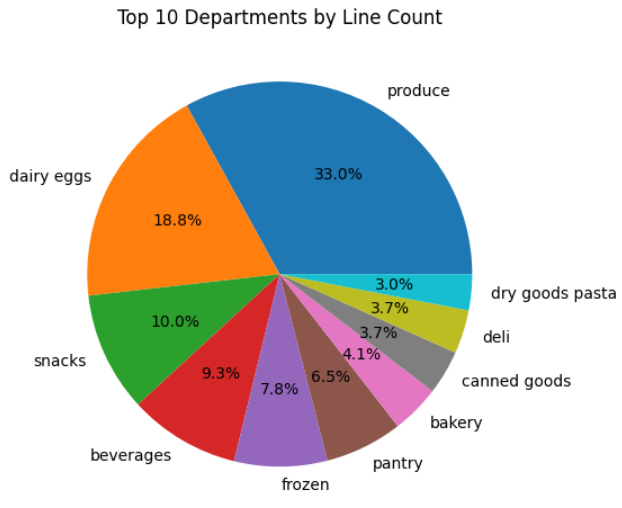
The final **master** table now describes each cart item along three axes at once:

* **Who** bought it and **when** (user\_id, weekday, hour, reorder lag)
* **What** it is (product name) and **where** it lives in the store (aisle, department)
* **How** it appeared in the basket (add\_to\_cart\_order, reordered flag)

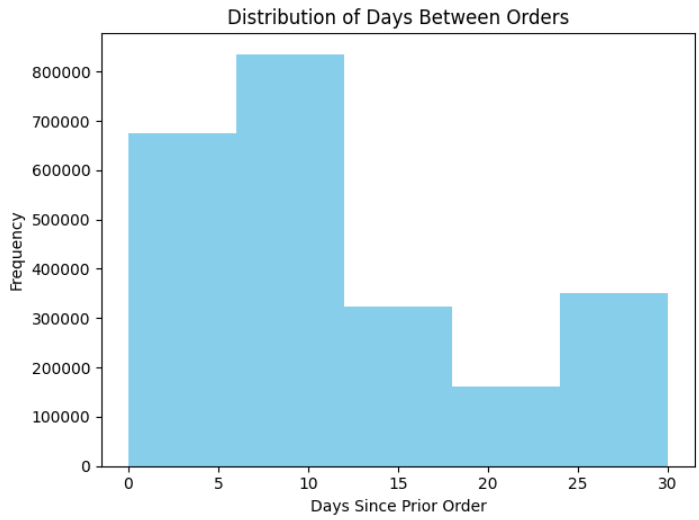
With all silos collapsed into this wide, analysis-ready view, we can jump straight into feature engineering, descriptive charts, or pattern-mining models that predict what a shopper will add to their basket next.

# **Exploratory Data Analysis**

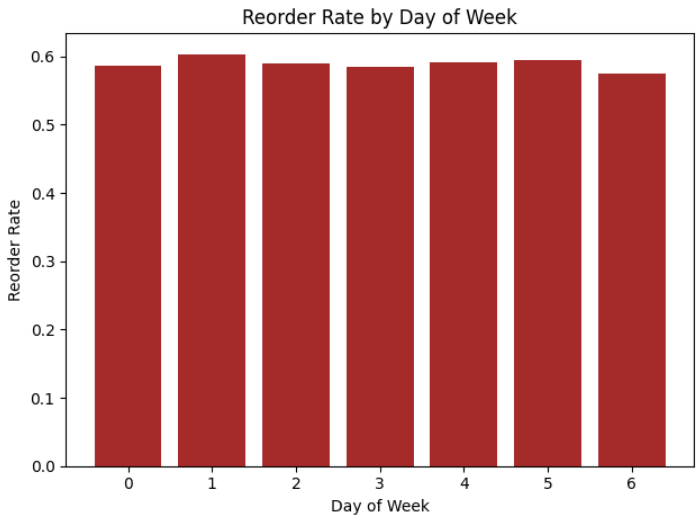
Exploratory Data Analysis concentrated on four quick but revealing visuals.  
**Pie chart of line-item share by department** showed a heavy skew: produce alone represents about a third of everything that goes through the Instacart cart, dairy-and-eggs tacks on another fifth, and snacks claim roughly ten percent. All remaining departments collectively sit below the ten-percent line. In plain terms, two aisles—fresh produce and the dairy case—dominate grocery-basket real estate; any layout move or bundle promotion that targets these zones will touch the majority of orders.



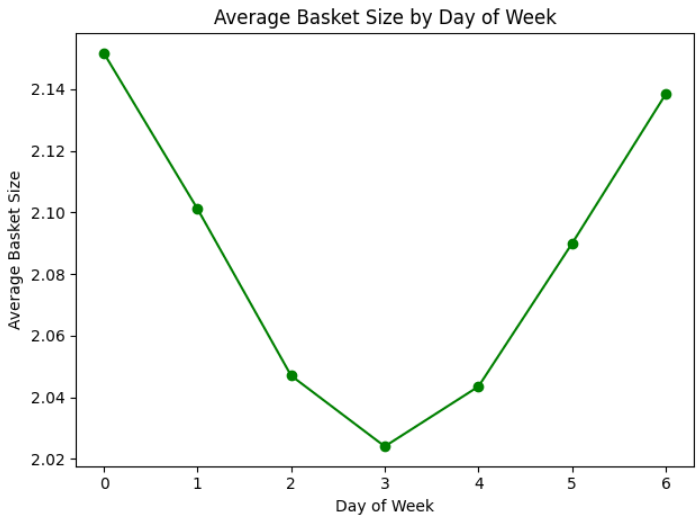
**Histogram of “days since prior order”** uncovered two clear shopping cadences. The tallest bars sit in the seven-to-ten-day window, signalling a weekly top-up rhythm for staples. A secondary bump between twenty-five and thirty days points to a monthly “stock-up” behaviour—think pantry refills or bulk household items. Those twin peaks translate directly into marketing triggers: a gentle weekly nudge for essentials and a monthly reminder for bigger hauls.



**Bar chart of reorder rate by day of week** was almost flat. Whether customers checked out on Monday or Saturday, about 58–60 % of the items in their baskets were repeats. That uniformity tells us that weekday alone isn’t a useful lever for timing promotions.



The **line plot of average basket size by day of week** painted a similarly steady picture. The mean order hovered around two items regardless of the calendar day, dipping only a few hundredths of an item mid-week. Basket size, therefore, is better segmented by its own distribution (small vs. large) than by weekday.



**Which two charts matter most?** The department pie and the reorder-lag histogram. The first pinpoints where volume and cross-sell upside live produce, dairy & eggs, snacks. The second tells us *when* shoppers are most receptive weekly for staples, monthly for pantry bulk. Together they give merchandising and marketing teams an immediate roadmap for placement, bundling and outreach.

## **Short-Cycle vs. Long-Cycle Reorder Behavior**

In addition to our core association and path models, we conducted a segmentation analysis based on **reorder interval behavior** to understand how basket composition changes across different shopping rhythms. Using the days\_since\_prior\_order field, we filtered the dataset into two cohorts:

* **Short-cycle users**: Customers who reorder between **7 to 10 days**, indicating habitual weekly top-up behavior.
* **Long-cycle users**: Customers with reorder intervals between **25 to 30 days**, suggesting bulk or stock-up behavior.

This segmentation was inspired by initial exploratory analysis, where a histogram of reorder gaps revealed two clear peaks—one at weekly cadence and another near the monthly mark. These behavioral anchors served as a natural basis for comparing basket content and association structures.

We began by extracting the **top 15 most frequently ordered products** for each group. The short-cycle group heavily favored **perishable staples** such as bananas, organic spinach, strawberries, milk, and avocados. This aligns with household restocking patterns, where items like fruits, greens, and dairy must be replenished frequently due to their limited shelf life.

In contrast, the long-cycle group showed a more varied mix: while bananas still ranked high, other top products included cucumbers, onions, and garlic—items that have longer shelf lives and are often used in bulk cooking or pantry storage. The diversity and durability of long-cycle orders reflect a different shopping intent, often driven by planning ahead or consolidating trips.

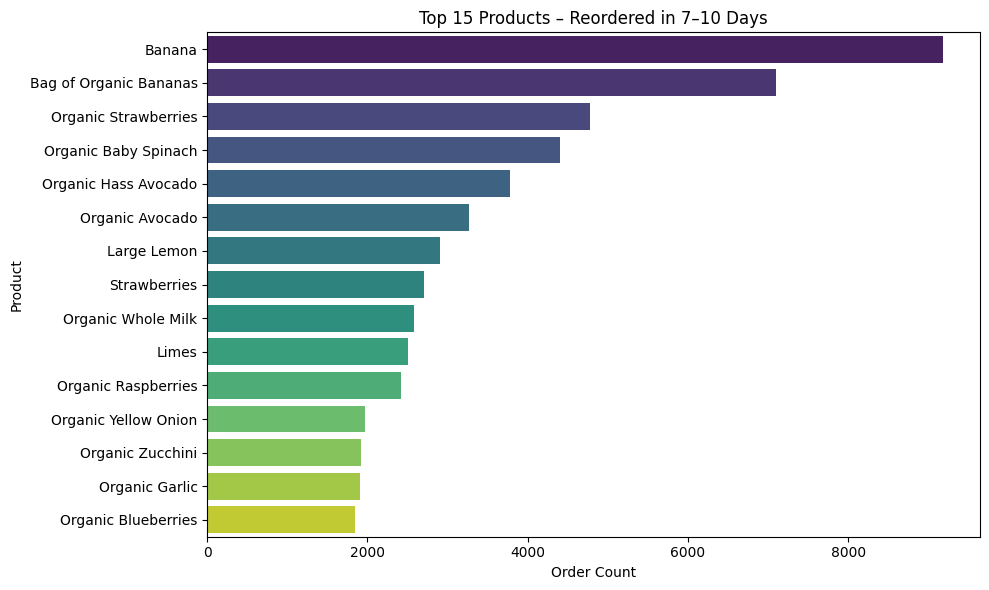
This segmentation revealed **clear shifts in consumer behavior based on time horizon**:

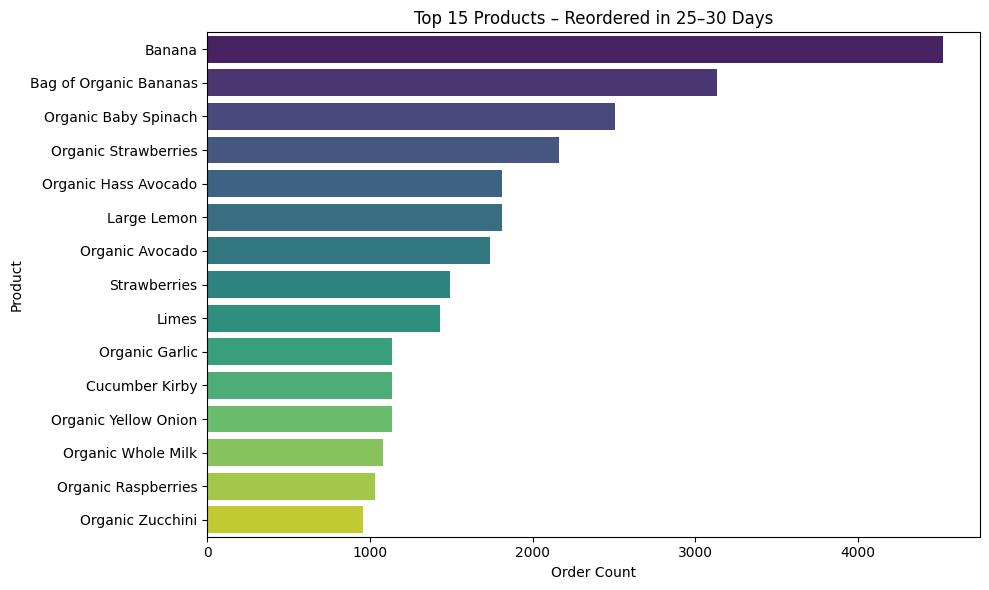
* **Short-cycle baskets** are smaller, more routine, and focus on immediate consumption.
* **Long-cycle baskets** tend to be larger, include pantry or multi-use items, and may reflect monthly grocery planning.

From a business perspective, these differences carry powerful implications:

* **Personalized reorder nudges** can be timed according to the customer’s natural cadence.
* Marketing campaigns could promote perishables for weekly shoppers and bulk deals for monthly shoppers.
* **Inventory planning** can be more tightly aligned with order frequency clusters, reducing waste or stockouts.

We also explored how association rules differ across these two segments. While short-cycle users showed strong co-purchases in breakfast and fresh categories (e.g., yogurt + berries), long-cycle users leaned toward cooking prep (e.g., garlic + onions, tomato sauce + pasta). These evolving combinations suggest that even the structure of customer affinities shifts with reorder timing.





Overall, segmenting users by reorder interval provided a behaviorally rich lens that enhanced our FP-Growth and cart path models. It allowed us to simulate realistic use cases like timed marketing nudges and targeted layout suggestions, grounded not just in what customers buy—but in when and why they buy it.

# **Models**

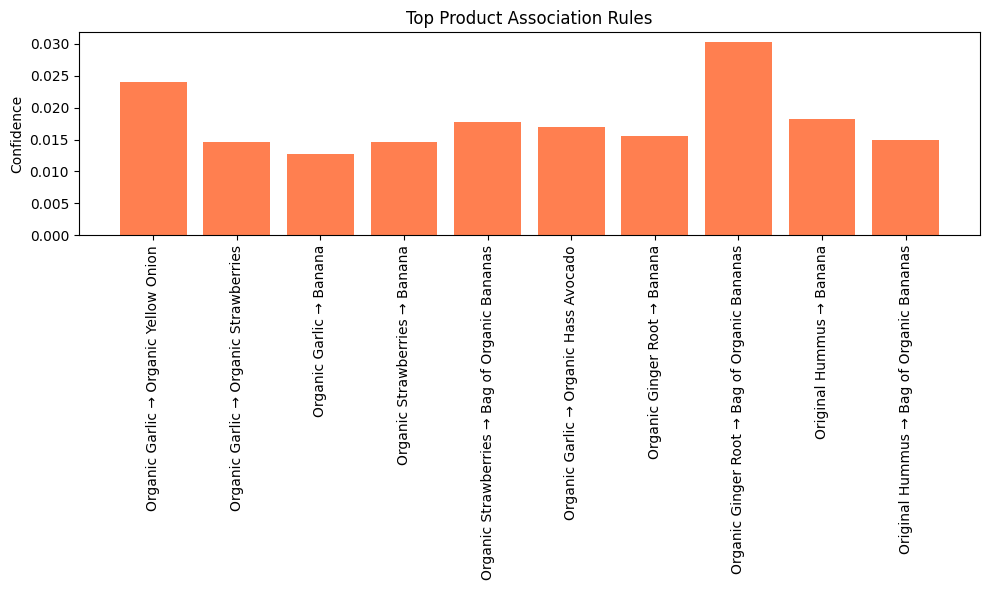
## **Product-Based Association Rules (FP-Growth)**

The first model in our basket analysis pipeline leverages Spark’s implementation of the FP-Growth algorithm to discover statistically significant co-purchase relationships between individual products. Each order in the dataset was treated as a transaction, with the **product\_name** column forming the itemset. We grouped items by **order\_id** using **collect\_set()** and fed these transactions into the FP-Growth model in PySpark with parameters minSupport = 0.0005 and minConfidence = 0.05.

This model uncovered intuitive yet actionable affinities. High-lift product pairs included *organic strawberries* and *bananas*, or *Greek yogurt* and *granola bars*—common breakfast or snack combinations. These findings confirmed that product-level co-occurrence can reveal implicit shopper intent, such as meal planning or habitual consumption. We sorted and filtered the rules by lift to prioritize associations with the strongest deviation from independence, which signals the most promising opportunities for bundling or recommendation.

The interpretability of this model was enhanced by visualizing the top product-product rules in a heatmap, using lift as the primary metric. These plots, rotated vertically to maximize readability (as recommended in our project feedback), helped reveal clusters of complementary items.

Despite its strengths, this model also had limitations: because it operates at the SKU level, results could be sparse due to the long-tail nature of grocery catalogs. Small spelling inconsistencies or infrequent items may have diluted some associations, making it necessary to complement this model with higher-level aggregation methods.



In practice, product-based association rules can be used for:

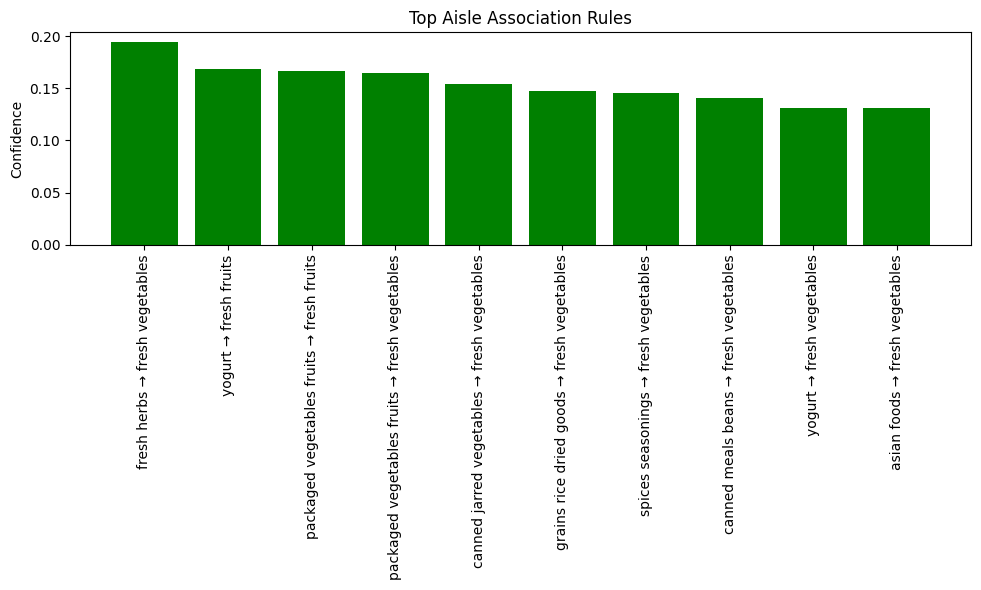
* **Real-time recommendation engines** ("People who bought X also bought Y")
* **Smart bundling** and **checkout upsell prompts**
* **Shelf adjacency decisions** in-store (to reduce shopper effort)

These rules serve as a foundation for personalizing experiences in online grocery environments, reducing friction, and lifting average basket size.

## **Aisle-Based Association Rules (FP Growth)**

To generalize beyond specific SKUs and reduce noise in the long tail, we implemented a second model at the aisle level. Here, we aggregated transactions into sets of aisle names (e.g., *fresh vegetables*, *snacks*, *yogurt*) and applied the same FP-Growth algorithm. This shift to category-level association improves robustness and broadens insights.

The resulting rules highlighted persistent behavioral themes. For instance, we observed strong associations between *yogurt* and *fresh fruit*, or *packaged vegetables* and *salad dressings*. These aisle-to-aisle co-purchases suggest routines like preparing breakfast or assembling a salad. A key advantage of this abstraction is its ability to surface behavior even when specific product combinations vary across customers.



Aisle-based rules are especially useful for:

* **Store layout optimization**: placing high-affinity aisles adjacently
* **Cross-promotions across departments** (e.g., yogurt discounts when purchasing berries)
* **Generalized bundle offers** that remain relevant across customer profiles

Visualizing these rules in a lift-based matrix heatmap provided a high-level overview of aisle affinities. Importantly, many high-lift pairs were cross-department, underscoring opportunities to break down traditional category silos in merchandising strategy.

By shifting from product granularity to categorical abstraction, this model improved both signal strength and managerial relevance—providing a flexible and scalable way to align store operations and digital menus with consumer behavior.

## **Reorder-Based Association Rules (FP-Growth)**

## **Cart Path Transition Modeling (Graph-Based)**

The fourth model took a sequence-oriented perspective, analyzing how items were added to the cart using the **add\_to\_cart\_order** variable. This allowed us to construct a directed graph of aisle-to-aisle transitions, effectively capturing the “path” a customer follows when assembling their basket.

We grouped items by order and sorted them by their cart position, then used Spark’s window functions (specifically lead() over a partitioned window) to extract consecutive aisle pairs. Aggregating across all orders resulted in a transition frequency matrix—a proxy for customer navigation flow, whether digital or in-store.

This model uncovered high-traffic transitions such as:

* *Dairy → Bakery*
* *Produce → Snacks*
* *Beverages → Frozen Foods*

These transitions suggest cognitive or behavioral patterns—e.g., customers grabbing breakfast essentials in sequence or moving from fresh goods to indulgent items. While digital shoppers don’t physically navigate a store, the order in which they select items often reflects real-life habits, shopping lists, or UI design flows.

This cart path model has several business applications:

* **Optimizing UI/UX** in mobile apps to mirror expected product sequences
* **In-store aisle arrangement** to align with natural item-pair flows
* **Reducing picker walk time** by pre-clustering high-transition aisles in fulfillment operations



From a data science perspective, the transition matrix serves as the foundation for **graph analytics**, such as PageRank or shortest-path algorithms, which can simulate optimal cart layouts or highlight critical “bridge” aisles.

This model complements the static associations of FP-Growth with **temporal and sequential context**, offering a deeper behavioral layer to guide strategic layout, bundling, and interface design.

# 

# **Appendix**

**Dataset Source**

H, M Yasser. “Instacart Online Grocery Basket Analysis Dataset.” *Kaggle*, 25 Jan. 2022, www.kaggle.com/datasets/yasserh/instacart-online-grocery-basket-analysis-dataset.

**Utilization of ChatGPT**

We used ChatGPT, an advanced AI tool, for a few aspects of our report and code development. This included debugging and interpreting code, which improved code reliability and efficiency. It also aided in enhancing our code documentation by adding structured headings and detailed comments to the code. Additionally, it helped in refining the grammar and structuring the overall layout of our project report, ensuring clarity and professionalism. This use of AI tools was disclosed to adhere to academic standards and to maintain transparency.