**Retail Customer Analytics Using PostgreSQL and MongoDB**

*Final Project Report*

**Executive Summary**

In today’s competitive grocery retail market, understanding customer behavior is key to driving growth. This project analyzed two years of grocery transaction data for 2,500 households, integrating both relational (PostgreSQL) and document-oriented (MongoDB) database approaches to derive insights. We cleaned and loaded the dunnhumby Complete Journey dataset, which includes household transactions, product details, customer demographics for 801 households, promotional campaigns, coupon redemptions, and in-store promotion data, into both database systems for a comparative analytics study. Our goal was to answer four strategic questions commonly posed by retail executives: (1) how customer spending trends change over time, (2) how demographics influence purchasing, (3) which product categories are growing or shrinking, and (4) whether direct marketing campaigns and in-store promotions boost customer engagement and sales.

We approached the analysis twice, first using a normalized schema in PostgreSQL and then using a denormalized document model in MongoDB, to evaluate the strengths and weaknesses of each platform for retail analytics. Key findings include: roughly half of households increased their spending from Year 1 to Year 2 while the others spent less, indicating a bifurcation where some customers became more engaged as others drifted away. Those who spent more tended to expand their purchases in staple grocery categories like pantry goods, produce, and dairy, and broaden into non-food departments such as personal care and fuel, signaling deeper loyalty and one-stop shopping behavior. Meanwhile, households that spent less cut back primarily on non-essentials like snacks and general merchandise while continuing to buy basic necessities like milk and bread, with some extreme cases dropping out entirely.

Demographic analysis revealed that household size, especially the presence of children, is the strongest driver of spend. Larger families spend far more in total, particularly on categories like milk, cereal, and baby products. Income and age had nuanced effects: higher-income households did spend more on premium products, but middle-income families with children contributed more to overall sales than the wealthiest segment, which underscores that family size often outweighs income in grocery spend. DINK households and single adults tended to spend a bit more on indulgences or premium categories compared to budget-conscious families, highlighting distinct segment behaviors. These insights suggest targeted marketing opportunities, for example promotions on bulk family essentials for larger households, versus premium or convenience items for smaller households.

We also found direct marketing campaigns had a measurable positive impact on customer engagement. Households that redeemed targeted coupons generally maintained or increased their spending relative to those who received no offers or ignored them. Across the customer base, those who participated in campaigns were less likely to decrease spend and often showed boosts in the categories of the coupons, indicating that well-designed mailer campaigns can drive incremental sales and loyalty. Analysis of in-store promotions showed that products featured on display or in the weekly ad tended to enjoy a lift in sales during the promotion period, confirming the value of merchandising efforts in driving short-term purchase spikes.

Technology-wise, the project highlighted complementary strengths of SQL and NoSQL for analytics. PostgreSQL’s structured schema and SQL queries excelled at complex joins and ensured data integrity through foreign keys, which was vital for combining transaction, product, and promotion tables reliably. MongoDB’s document model allowed us to answer certain questions with fewer lookups. For example, computing basket totals and item counts in MongoDB was very fast by leveraging embedded arrays, avoiding the multi-table join that PostgreSQL required. However, for multi-faceted analyses like tying transactions to product attributes or large promotional data, PostgreSQL’s set-based operations were more straightforward, whereas MongoDB needed careful pipeline design and extensive indexing to achieve similar performance.

In summary, PostgreSQL provided robustness and simplicity for cross-table analytics, while MongoDB offered speed and flexibility for aggregated, self-contained data like basket-level metrics. We recommend a hybrid strategy: use PostgreSQL as the source of truth and for heavy relational crunching where data integrity and complex joins matter, and use MongoDB for fast analytics on pre-joined views that benefit from quick one-to-many aggregations. The final section provides concrete recommendations on which analytical tasks suit each system best and how to leverage both in a complementary fashion.

**Introduction**

Retailers are constantly looking to analyze customer behavior to tailor marketing strategies and improve sales. In grocery retail, understanding who your customers are, how their spending changes, what products they buy, and whether marketing efforts keep them engaged can inform everything from inventory management to personalized promotions. In this project, I set out to explore these questions using a rich transactional dataset and to evaluate how two different database technologies, PostgreSQL and MongoDB, support such analytics. By conducting the same analyses on both systems, we can identify practical trade-offs in data modeling, query patterns, and performance, thereby guiding data architects and business stakeholders on the best tool for each job.

**Business Questions:**

* Customer Spending Trends: Are customers spending more or less over time? What are the weekly or seasonal patterns in total sales?
* Demographic Influences: How do customer demographics like family size and income affect how much and how often they shop?
* Category Growth: Which product categories are growing the fastest, and which are stagnating or declining, especially among customers who increase or decrease their spend?
* Marketing Campaign Impact: Do coupon mailers and in-store promotions lift sales or basket sizes? Can we see differences between households who redeem offers and those who do not?

**Data Source:** We used the dunnhumby “The Complete Journey” dataset, which provides a comprehensive view of household-level grocery shopping over two years. This dataset covers 2,500 households who are frequent shoppers at a grocery chain, recording every item purchased over a 104-week period. For 801 households, the data also includes demographic information and a history of direct marketing campaigns they received. Additionally, it includes reference tables for all products sold, detailed logs of coupon offerings and redemptions, and a large table of in-store promotions indicating whether a product was on display or featured in a weekly flyer for each store-week. The breadth of this data allowed us to connect customer profiles to purchases and marketing exposures, which is ideal for holistic retail analysis.

**Data Volume and Structure:**

* Household Demographics: 801 rows. One row per household with attributes like age group, marital status, income band, homeowner status, household size, and family composition. Primary key: household\_key.
* Transaction Line Items: about 1.43 million rows after filtering. Each record logs one purchased product on a basket, with household\_key, basket\_id, product\_id, quantity, sales\_value, discount fields, store\_id, and day/week.
* Products: 92,353 rows. Each product\_id has department, commodity, sub-commodity, brand, manufacturer, and size.
* Campaign Description: 30 rows. Each campaign has id, type, and start\_day/end\_day.
* Campaign Assignments: about 4,213 rows after filtering. Each row ties a household to a campaign (composite key).
* Coupons: 119,384 rows after de-duplication. Each row ties a coupon\_upc and campaign to an eligible product\_id.
* Coupon Redemptions: 1,856 rows after filtering. Each row logs a household redeeming a specific coupon in a specific campaign on a specific day.
* Causal Promotions: about 36.8 million rows. Each row indicates if a product had a promotional display or mailer in a given store and week.

All tables link through common keys: household\_key, product\_id, campaign, and coupon\_upc. The normalized relational model reflects third normal form so analytic joins behave predictably.

**Data Preparation and ETL Process**

We built a single ETL pipeline in Python that we executed twice, once for PostgreSQL and once for MongoDB, using identical cleaning logic to ensure a fair comparison.

* **Loading:** We read the raw CSVs with pandas and normalized column names. We aligned fields like demographic classifications to readable names.
* **Cleaning invalid references:** Because only 801 households have demographics, we filtered out transactions, campaign assignments, and coupon redemptions that referenced other households. This created a consistent universe for demographic analyses and mirrored the effect of enforcing foreign keys.
* **De-duplication:** We dropped exact duplicate coupon rows so that each (campaign, coupon\_upc, product\_id) is unique.
* **PostgreSQL schema:** We created normalized tables with primary keys and foreign keys, then bulk loaded the cleaned data with COPY in large batches. Indexes on keys and common joins prepared the database for analytics.
* **MongoDB design:** We designed collections around access patterns.
  + Households: one document per household with demographic attributes.
  + Products: one document per product with denormalized category descriptors to avoid joins when slicing by category.
  + Baskets: one document per basket with an embedded items array that contains product\_id, quantity, sales\_value, and discounts.
  + Campaigns: one document per campaign with attributes and an embedded array of targeted households.
  + Coupons: one document per (campaign, coupon\_upc) with an array of eligible product\_ids.
  + CouponRedemptions: one document per redemption.
  + CausalPromotions: one document per product-store-week with display and mailer flags.

We inserted to MongoDB with chunked insert\_many and sanitized data to native Python types. We created indexes that align with the analytical queries, such as compound indexes on join keys and multikey indexes on items.product\_id.

**Analytical Approach and Key Insights**

**Customer Spending Trends Over Time**

We aggregated household spend by year and compared Year 1 to Year 2. The result showed a near split between households that increased and those that decreased their spend. Weekly sales aggregated across all households were steady with mild holiday peaks. The distribution of basket totals and item counts was highly skewed. The median basket was modest, but there was a long tail of large stock-up baskets. Inter-purchase gaps suggested a weekly cadence for many households, with most making at least a monthly trip.

**Implication:** Focus retention efforts on the declining group with targeted offers, and cultivate loyalty among increasers by reinforcing staple categories and cross-selling into adjacent categories.

**Category Growth and Decline**

We compared Year 2 vs Year 1 spend by department and sub-commodity for the two household segments.

* **Increasing-spend households:** Grew spend in staples such as Grocery, Produce, and Dairy, and expanded into non-food categories like Drug/GM and Fuel. This indicates deeper reliance on the store as a one-stop shop.
* **Decreasing-spend households:** Reduced spend on discretionary categories such as snacks and general merchandise, while maintaining essentials. Some churned entirely.

**Implication:** Invest in staple value for loyalty, promote cross-department offers for increasers, and use value-led promotions to re-engage decliners in categories they cut first.

**Demographic Influences on Spending**

Family composition dominated spend. Larger households, especially with children, drove the highest total spend and favored categories like milk, cereal, and baby products. DINK and single households showed higher per-capita discretionary spend on premium or indulgence categories. Higher income correlated with premium choices but not necessarily with the largest total sales contribution, where middle-income families with children dominated.

**Implication:** Segment by life stage and family size. Offer bulk deals and staple incentives to family segments, and premium or convenience offers to small households.

**Impact of Marketing Campaigns and Promotions**

Coupon redeemers tended to maintain or increase spend more than non-redeemers, indicating that engagement with offers correlates with retention. Target vs control comparisons for specific campaigns showed a modest uplift in basket totals during campaign windows for targeted households. In-store displays and mailer features produced clear short-term lifts in item sales during the promotion weeks.

**Implication:** Continue direct marketing and refine targeting to maximize redemption. Pair campaign coupons with in-store displays for synergy. Track uplift at the category level to allocate promotion budgets efficiently.

**PostgreSQL vs MongoDB – Analytical Workflows Compared**

**Schema and query expression:**

* PostgreSQL uses normalized tables and explicit foreign keys. Complex analyses often require multi-table joins but are concise to express in SQL.
* MongoDB leverages document shapes. Embedded arrays reduce or eliminate joins for one-to-many aggregations. Aggregation pipelines can be verbose, but when data is modeled around access patterns, queries are short and fast.

**One-to-many rollups:**

* PostgreSQL: group by basket\_id over a large fact table.
* MongoDB: compute basket totals and item counts by projecting $sum over the embedded items array. MongoDB is usually faster here because the data is pre-joined in the document.

**Many-to-many and high-cardinality joins:**

* PostgreSQL: strong and predictable with indexes and the optimizer, especially for joins like transactions to promotions on composite keys.
* MongoDB: requires careful pipeline design. Pre-aggregation to reduce cardinality before $lookup and compound indexes make a big difference. It can match performance but needs more modeling effort.

**Indexing:**

* PostgreSQL: btree indexes on foreign keys and composite indexes on heavy joins.
* MongoDB: compound indexes on lookup keys and multikey indexes on arrays such as items.product\_id. Index strategy in both systems often determines success more than the engine choice.

**Integrity and governance:**

* PostgreSQL enforces integrity at write time. Cleaning happens once and constraints keep data consistent.
* MongoDB places the burden on ingestion and validation but is flexible when adding or reshaping attributes.

**Developer productivity:**

* SQL is succinct and easy to audit for multi-table analytics.
* MongoDB pipelines shine when queries align with document boundaries. For deep relational logic, SQL is simpler to read and maintain.

**Performance summary:**

* Use MongoDB for document-centric rollups and fast basket or household summaries.
* Use PostgreSQL for multi-entity analytics, many-to-many correlations, and where window functions or complex filters are needed.
* With proper indexes and modeling, both systems meet performance targets for this dataset.

**Conclusion and Recommendations**

This project demonstrated that both PostgreSQL and MongoDB can successfully handle retail customer analytics, but each excels in different areas. By leveraging the two in tandem, a business can enjoy the reliability of relational data and the agility of document-based queries.

**Our recommendations:**

* **Complex multi-table analytics:** Prefer PostgreSQL. Examples include category growth across the full hierarchy, promotion lift across large product sets, and target vs control campaign analyses that demand precise joins and filtering.
* **One-to-many document rollups:** Prefer MongoDB. Examples include basket totals, basket size distributions, recent household spend summaries, and other metrics that live inside a single natural aggregate.
* **System of record and governance:** Keep PostgreSQL as the master data store for products, households, and transactions where integrity is essential.
* **Flexibility and speed for applications:** Use MongoDB for serving aggregated views that power real-time dashboards or microservices, such as precomputed household profiles or campaign snapshots.
* **Hybrid architecture:** Maintain a normalized warehouse in PostgreSQL and publish curated aggregates to MongoDB collections tailored to the most frequent analytical queries.
* **Index deliberately:** Treat index design as a first-class part of analytics. In MongoDB, prioritize compound indexes that match $match and $lookup patterns. In PostgreSQL, index join keys and consider materialized views for expensive recurring queries.
* **Promotions and campaigns:** Continue direct marketing and displays. Measure uplift per campaign and category in PostgreSQL for rigor, and expose near real-time engagement metrics in MongoDB for operational action.

**Executive Recommendations (Summary):**

* Use PostgreSQL for year-over-year spend, cohort retention, category growth across the full catalog, and rigorous target vs control campaign evaluations.
* Use MongoDB for per-basket and per-household rollups, real-time dashboards that need fast retrieval of aggregated metrics, and flexible application-driven analytics.
* Keep relational integrity in PostgreSQL as the source of truth. Publish selectively to MongoDB for speed where the analytics fit document boundaries.
* Invest in indexes in both systems aligned with the top analytical questions.
* Pair coupon campaigns with in-store displays to maximize lift, and target offers based on family size and life stage to improve redemption and retention.