

RevoGrocers Sales Performance Analysis

Santa Riyanti T
28 November 2025

RevoU FSDA Batch OCT25



COMPANY OVERVIEW

RevoGrocers is a fictional grocery retail business that operates in multiple locations, offering a diverse range of grocery product to customers. The company aims to optimize sales strategies, enhance customer experience, and increase revenue by leveraging data-driven decision making.

Problem Statement

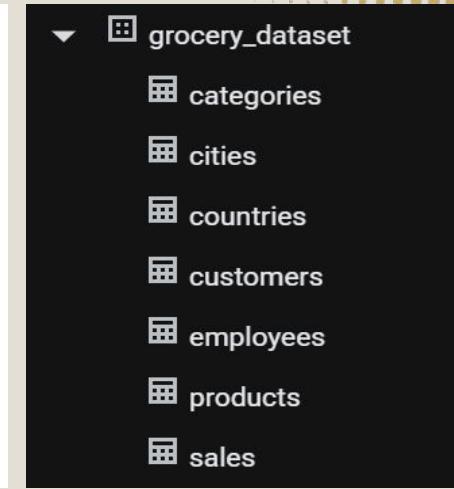
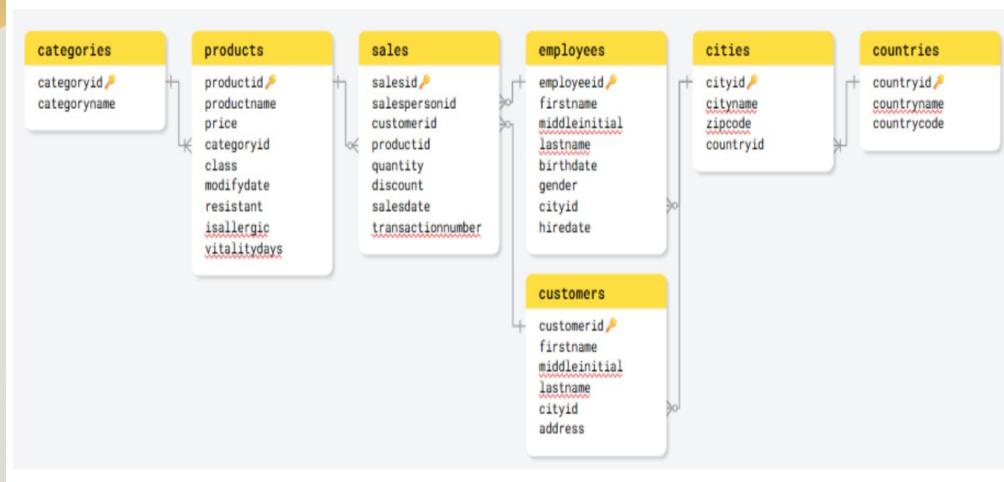
How to analyze sales performance across different product categories, that can extract insights to answer business critical questions that drive decision making in retail operations ?

Objectives

Identify top performing product categories, analyze factors driving revenue (units solds vs customer count), and evaluate pricing strategies and their impact on sales by Tools Used Google BigQuery for running SQL.

DATASET OVERVIEW

[link](#)



DISCLAIMERS :

- This analysis is based on a publicly available Kaggle dataset and does not reflect real-world financial or business insights.
- RevoGrocers is a fictional entity and the results presented here are purely for educational purposes.
- The queries and insights generated in this assignment should be personalized analysis results.

QUESTION 1 Identify the product category that generates the highest revenue after discount [link](#)

```
SELECT
    categoryname as categories
    , SUM(s.quantity * p.price * (1 - s.discount)) AS
total_revenue_after_discount
FROM fsda-sql-01.grocery_dataset.sales s
JOIN fsda-sql-01.grocery_dataset.products p
    ON s.productid = p.productid
JOIN fsda-sql-01.grocery_dataset.categories c
    ON p.categoryid = c.categoryid
GROUP BY 1
ORDER BY 2 DESC
LIMIT 1
```

Row	categories	total_revenue_after_discount
1	Confections	556930717.3468616

The goal of this query is to identify which product category generates the highest revenue after discount. Since **the ERD shows that the product price, discount, and purchases quantities are stored across multiple tables (sales, products, and categories), the query needs to connect them and calculate the adjusted revenue per category (use JOIN).**

After running the query, the product category **Confections represents the most profitable category after applying customer discount, with a total revenue of over 556 million.** This category drives the largest share of business revenue and has the strongest financial performance even after price reductions.

QUESTION 2 Assess the relation between revenue after discount and total units sold for each category [link](#)

```
SELECT
    categoryname as categories
    , SUM (s.quantity) AS total_units_sold
    , SUM (s.quantity * p.price * (1 - s.discount)) AS
revenue_after_discount
FROM fsda-sql-01.grocery_dataset.sales s
JOIN fsda-sql-01.grocery_dataset.products p
    ON s.productid = p.productid
JOIN fsda-sql-01.grocery_dataset.categories c
    ON p.categoryid = c.categoryid
GROUP BY 1
ORDER BY 3 DESC
```

- This query was selected because the analytical objective is to assess the relationship between Total Revenue after discount and total unit sold at the product category level. Since the **necessary data is stored across multiple tables (sales, products and categories)**, **the query requires several JOIN operation** on them.
- By performing both joins, the query can start with a sales record, **find the corresponding product detail (price)**, and then find that product's category name. This allow the **aggregation (SUM) of total units sold and revenue after discount to be grouped correctly by category name** as requested by the original prompt (for each category)

Row	categories	total_units_sold	revenue_after_discount
1	Confections	11078474	556930717.3468616
2	Meat	9719292	492888844.69468868
3	Poultry	9159847	440025564.96562117
4	Cereals	8735296	427393431.91017866
5	Snails	7199409	372084885.20758617
6	Beverages	7393693	366515024.00501525
7	Produce	8174673	362861133.51715457
8	Dairy	6815143	354358156.76889551
9	Seafood	6996152	330527987.46767879
10	Grain	5433152	323879126.65000933
11	Shell fish	6983457	299598294.02995741

The output table provides clear insights about the relation between revenue after discount and total units sold for each category :

- **Confections ranks as the top-performing category (total units sold 11.078.474 units and revenue after discount 556 million)**, leading both in total units sold and revenue after discount. This suggests strong customer demand and effective pricing despite discounts.
- **Meat and Poultry follow closely, with relatively high revenue and unit sales.** These categories likely consist of higher-priced essential products, contributing significantly to total revenue.
- **Cereals, Snails, Beverages, and Produce fall into the middle tier.** They show steady revenue and moderate sales volume, indicating consistent demand but possibly lower pricing or higher discount usage.
- **Dairy and Seafood generate lower revenue despite moderate units sold.** This may indicate lower pricing, high discount rates, or less frequent purchases.
- **Grain and Shell fish are the weakest performers**, ranking last in both metrics. This suggests low consumer demand, limited product availability, or pricing that does not drive high sales volume.

QUESTION 3 Find the relation between revenue after discount and the number of unique customers for each product category [link](#)

```
SELECT
    categoryname AS categories
    , COUNT (DISTINCT s.customerid) AS unique_customers
    , SUM (s.quantity * p.price * (1 - s.discount)) AS
revenue_after_discount
FROM fsda-sql-01.grocery_dataset.sales s
JOIN fsda-sql-01.grocery_dataset.products p
    ON s.productid = p.productid
JOIN fsda-sql-01.grocery_dataset.categories c
    ON p.categoryid = c.categoryid
GROUP BY 1
ORDER BY 3 DESC
```

The chosen SQL syntax is designed to address the analytical goal by using :

- **Selecting and Aggregating Data** (Use Distinct and SUM)
- **Joining Tables**, the query requires data from three different tables (sales, products and categories), join operation are used to link related records.
- **Grouping and Ordering** (For Grouping, it instructs the database to perform the count and SUM calculation for each distinct categoryname and For Ordering, making the final results table in descending based on column revenue after discount.

Row	categories	unique_customers	revenue_after_discount
1	Confections	98743	556930717.3468616
2	Meat	98701	49288844.69468868
3	Poultry	98679	440025564.96562117
4	Cereals	98651	427393431.91017866
5	Snails	98376	372084885.20758617
6	Beverages	98424	366515024.00501525
7	Produce	98601	362861133.51715457
8	Dairy	98308	354358156.76889551
9	Seafood	98334	330527987.46767879
10	Grain	97335	323879126.65000933
11	Shell fish	98338	299598294.02995741

The results table provides clear **insights into which categories drive the most revenue and customer engagement**:

- **Confections Lead Revenue** : The Confections category generated the highest revenue by a significant margin (556,930,717..).
- **Meat is Second** : The Meat category is the second highest in revenue (492,888,844...).
- **Consistent Customer Base** : A key insight is the remarkable consistency in the number of unique customers across almost all categories listed. **Most categories have a unique customer count very close to 98,000** (e.g., Confections 98743, Meat 98701, Poultry 98679).

QUESTION 4 Calculate the average price unit for each product category catalogue [link](#)

```
SELECT
    categoryname as categories
    , AVG(p.price) as avg_unit_price
FROM fsda-sql-01.grocery_dataset.products p
JOIN fsda-sql-01.grocery_dataset.categories c
    ON p.categoryid = c.categoryid
GROUP BY 1
ORDER BY avg_unit_price DESC
```

Row	categories	avg_unit_price
1	Grain	61.40394642...
2	Dairy	53.55683142...
3	Snails	53.19933243...
4	Meat	52.27465199...
5	Confections	51.84993157...
6	Beverages	51.04368421...
7	Cereals	50.41638888...
8	Poultry	49.46088297...
9	Seafood	48.66665555...
10	Produce	45.79683333...
11	Shell fish	44.27278333...

This query performs the following steps :

- It **joins the products table with the categories table** based on categoryID.
- It **uses the AVG() function to calculate the average unit price for each category**.
- It **applies GROUP BY categoryname** to ensure the calculation is performed per category.
- It orders the results from the highest to the lowest average price, **use ORDER BY avg_unit_price DESC**.

The analysis of the average unit price across product categories shows :

- That **Grain has the highest average unit price, followed by Dairy, Snails, and Meat**. These categories likely consist of premium or high-cost items, which may contribute more revenue per unit sold.
- On the other hand, **categories such as Shell fish, Produce, and Seafood show the lowest average unit prices**. These lower-priced categories may rely more on higher sales volume to generate significant revenue.
- This pricing pattern indicates a potential tiered pricing structure within the product catalog, where certain categories are positioned as premium while others serve as more affordable options.

QUESTION 5 Evaluate the relation between the avarage price per unit and the number of buyers (unique customers) per category [link](#)

```
SELECT
    categoryname AS categories
    , AVG(p.price) AS avg_unit_price
    , COUNT(DISTINCT s.customerid) AS unique_buyers
FROM fsda-sql-01.grocery_dataset.sales s
JOIN fsda-sql-01.grocery_dataset.products p
ON s.productid = p.productid
JOIN fsda-sql-01.grocery_dataset.categories c
ON p.categoryid = c.categoryid
GROUP BY 1
ORDER BY 2 DESC
```

The chosen SQL query is designed to evaluate the relationship between the average unit price and the number of unique buyers (unique customers) for each product category.

1. **Selecting and Aggregating Data** (by SELECT categoryname AS categories, use aggregate function AVG, and count only unique customer/ COUNT DISTINCT).
2. **Joining tables** because data is stored across three normalized tables (sales, products, and categories).
3. **Grouping aggregate the AVG and Count** metrics for each **distinct categoryname** and **Ordering** the final results table by **descending order based on the average unit price (avg_unit_price)** .

The provided results table offers clear insights between the average price per unit and the number of buyers (unique customers) per category across different product types :

- Expensive Categories** : Grain are the most expensive category on average, with an average unit price of approximately 61.43, followed closely by Dairy at 53.61.
- Affordable Categories** : Shell fish, Produce, and Seafood have the lowest average unit prices.
- Consistent Customer Reach** : A primary insight is the extremely consistent number of unique buyers across nearly all categories. Almost every category has a unique buyer count between 97,000 and 98,700 , regardless of the price point.
- Low-Price** : While most categories have similar buyer counts, the very cheapest category, Shell fish also has the lowest number of unique buyers (98338) .

Row	categories	avg_unit_price	unique_buyers
1	Grain	61.43254359...	97335
2	Dairy	53.61147548...	98308
3	Snails	53.28068015...	98376
4	Meat	52.31211525...	98701
5	Confections	51.81190322...	98743
6	Beverages	51.12591015...	98424
7	Cereals	50.43802735...	98651
8	Poultry	49.50921884...	98679
9	Seafood	48.67796017...	98334
10	Produce	45.78994061...	98601
11	Shell fish	44.23266616...	98338

QUESTION 6 Which categories contribute the most to overall revenue after discount (percentage)

[link](#)

```
WITH revenue_by_category AS (
    SELECT
        c.categoryname
        , SUM(s.quantity * p.price * (1 - s.discount)) AS revenue_after_discount
    FROM fsda-sql-01.grocery_dataset.sales s
    JOIN fsda-sql-a01.grocery_dataset.products p
        ON s.productid = p.productid
    JOIN fsda-sql-01.grocery_dataset.categories c
        ON p.categoryid = c.categoryid
    GROUP BY 1
),
total_revenue AS (
    SELECT
        SUM(revenue_after_discount) AS total_revenue
    FROM revenue_by_category r
)
SELECT
    r.categoryname
    , r.revenue_after_discount AS total_revenue
    , ROUND ((revenue_after_discount / total_revenue) * 100, 2) AS percentage_contribution
FROM revenue_by_category r
CROSS JOIN total_revenue t
ORDER BY percentage_contribution DESC
```

The specific syntax selection is explained as follows :

1. **WITH (Common Table Expressions / CTE)**
Used to organize the query into logical steps that are easier to read and understand.
2. **revenue_by_category** calculates the total revenue after discounts for each category using the SUM() aggregate function and GROUP BY.
3. **total_revenue** calculates the overall total revenue from the results of the first CTE. JOIN used to link the sales, products, and categories tables based on productid and categoryid to access the necessary data (category name, quantity, price, discount).
4. **Main SELECT** combines per-category revenue data with the total revenue using a CROSS JOIN (since the total revenue is a single row). ROUND() used to format the percentage contribution to two decimal places for neatness. ORDER BY percentage_contribution DESC, sorts the final results from the largest to smallest percentage contribution as per the query's objective.

Row	categoryname	total_revenue	percentage_contribution
1	Confections	556930717.3...	12.87
2	Meat	492888446.0...	11.39
3	Poultry	440025564.9...	10.17
4	Cereals	427393431.9...	9.88
5	Snails	372084885.2...	8.6
6	Beverages	366515024.0...	8.47
7	Produce	362861133.5...	8.39
8	Dairy	354358156.7...	8.19
9	Seafood	330527987.4...	7.64
10	Grain	323879126.6...	7.48
11	Shell fish	299598294.0...	6.92

This table displays the **ranking of revenue contributions from eleven different product categories to the overall total revenue**. The data has been sorted in descending order based on the percentage contribution column, making it easy to see which categories are most significant in generating revenue.

Results Analysis :

- **Top Rank : The Confections** category is ranked highest, contributing 12.87% of the total revenue.
- **Lowest Rank : The Shell fish** category is ranked last among the displayed data, contributing 6.92% of the total revenue.

QUESTION 7 Which product categories have the highest repeat purchase rate [link](#)

The SQL syntax **uses Common Table Expressions (CTE)** to calculate the repeat purchase rate percentage per product category.

1. Use of WITH (Common Table Expressions)

The WITH clause, also known as a CTE, is used to create a temporary result set that can be referenced within a subsequent main query. It is chosen for Breaking down a complex query into smaller, named parts (customer_purchases, categoryStats), making it easier to understand and manage.

2. Use three logical stages of the query

• Customer_purchases CTE

To identify the number of unique purchases per customer and category.

Uses JOIN to connect the sales (sales), products (products), and categories (categories) tables. COUNT(s.salesid) is used to count the number of transactions (purchases) for each category and customer combination, then grouped (GROUP BY 1, 2, 3).

• CategoryStats CTE

To calculate the total unique customers and the count of repeat customers for each category.

COUNT(DISTINCT customerid) Syntax : Calculates the total number of unique customers who have ever purchased in that category.

COUNT(DISTINCT CASE WHEN ... THEN ... ELSE NULL END)

Syntax : This is an efficient way to conditionally count repeat

customers. It counts only the unique customers who had a purchase_count > 1 from the previous CTE. Use HAVING

COUNT(DISTINCT customerid) > 1, to filter the aggregate.

• Main Query

To calculate the percentage and present the results.

CAST(... AS FLOAT64) Syntax : This is crucial to ensure division results in a decimal number (floating point) rather than integer division, which is necessary to calculate percentages accurately.

WHERE total_customers > 0 : Ensures no division by zero occurs.

ORDER BY 4 DESC : Sorts the results by the highest repeat purchase rate percentage to easily identify top-performing categories.

```
WITH customer_purchases AS (
    SELECT
        p.categoryid
        , c.categoryname
        , s.customerid
        , COUNT(s.salesid) AS purchase_count
    FROM fsda-sql-01.grocery_dataset.sales s
    JOIN fsda-sql-01.grocery_dataset.products p
        ON s.productid = p.productid
    JOIN fsda-sql-01.grocery_dataset.categories c
        ON p.categoryid = c.categoryid
    GROUP BY 1,2,3
),
categoryStats AS (
    SELECT
        categoryid
        , categoryname
        , COUNT(DISTINCT customerid) as total_customers
        , COUNT(DISTINCT CASE WHEN purchase_count > 1 THEN customerid ELSE NULL END) as
repeat_customers
    FROM customer_purchases
    GROUP BY 1,2
    HAVING COUNT(DISTINCT customerid) > 1
)
SELECT
    categoryname
    , total_customers
    , repeat_customers
    , CAST(repeat_customers AS FLOAT64) / CAST(total_customers AS FLOAT64) * 100 as
repeat_purchase_rate_percent
FROM categoryStats
WHERE total_customers > 0
ORDER BY 4 DESC
```

Here are some insights from the SQL syntax output data :

- **"Confections" Category Leads** : The "Confections" category has both the highest total number of customers and the **highest repeat purchase rate at 99.85%**. This makes it the most successful category in terms of loyalty and customer base. This indicates extraordinarily high customer loyalty across all categories or perhaps suggests the essential or consumable nature of the products that require routine purchases.
- **"Grain" Category Has the Lowest Rate** : The "Grain" category has the **lowest repeat purchase rate (93.68%) and the lowest total number of customers**.

Row	categoryname	total_customers	repeat_customers	repeat_purchase_rate_percent
1	Confections	98743	98598	99.853154147635777
2	Meat	98701	98318	99.61195935198225
3	Poultry	98679	98122	99.435543530031723
4	Cereals	98651	97867	99.205279216632363
5	Produce	98601	97550	98.934087889575167
6	Beverages	98424	96679	98.22705844103065
7	Snails	98376	96324	97.914125396438152
8	Seafood	98334	96138	97.766794801391171
9	Shell fish	98338	96054	97.67739836075576
10	Dairy	98308	95677	97.323717296659467
11	Grain	97335	91184	93.6805876611702

QUESTION 8 Summarize overall findings, regarding the sales performance across product categories (key trends and anomalies)

Key Trends Insight

- **Volume as the Primary Revenue Driver** : There is a very strong correlation between the total_units_sold column and revenue_after_discount. The category with the highest units sold (**Confections**) also generates the highest revenue. This indicates that the current business strategy relies heavily on high sales volume to drive revenue.
- **Solid Performance in the Protein Segment** : Meat, Poultry, and Seafood categories collectively contribute a significant and consistent share of revenue, reflecting stable market demand for these products.
- **High Retention Across Categories** : Repeat purchase rates exceed 93%, with some near 100%, showing strong customer loyalty.

Anomalies Insight

- **Customer Metrics vs. Revenue Anomaly** : The most significant anomaly lies in the **inconsistency between customer count and revenue**. Although all categories have nearly the same number of unique customers, the revenue gap between the top category (Confections) and the bottom category (Shellfish) exceeds 250 million. This indicates that revenue differences are not primarily driven by customer volume, but rather by the average transaction value per customer or significantly higher purchase frequency in certain categories (such as Confections).
- **Category Ranking of 'Snails'** : The Snails category ranks 5th with relatively high revenue (around 372 million), surpassing more common categories such as Beverages, Produce, and Dairy. This may suggest a **niche market segment characterized by high per-unit product prices or exceptionally strong demand from a loyal customer base**.
- **Efficiency of the Grain Category** : The Grain category records the **lowest sales volume in the list (5.43 million units)**, yet its **revenue (around 324 million)** is relatively high compared to its volume. This may indicate higher per-unit prices or better margins than some other categories, such as Shellfish, which sells more units but generates slightly lower revenue.

QUESTION 9 Find the cumulative amount of transaction of the top user (*user with highest transaction value*) [link](#)

```
WITH customer_total_sales AS (
    SELECT
        s.customerid
        , SUM(s.quantity * p.price) AS total_spent
    FROM fsda-sql-01.grocery_dataset.sales s
    JOIN fsda-sql-01.grocery_dataset.products p
        ON s.productid = p.productid
    GROUP BY 1
),
top_spenders AS (
    SELECT
        customerid
        , total_spent
        , RANK() OVER (ORDER BY total_spent DESC) AS customer_rank
    FROM customer_total_sales
)
SELECT
    s.saleid
    , s.customerid
    , s.salesdate
    , (s.quantity * p.price) AS transaction_amount
    , SUM(s.quantity * p.price) OVER (PARTITION BY s.customerid ORDER BY s.saleid,
s.salesdate ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cumulative_amount
FROM fsda-sql-01.grocery_dataset.sales s
JOIN fsda-sql-01.grocery_dataset.products p
    ON s.productid = p.productid
WHERE s.customerid IN (SELECT customerid FROM top_spenders WHERE customer_rank = 1)
ORDER BY 2,3
```

Relevant schema for this query, includes the **Customers, Sales, and Products tables**.

Step 1 : Determine the total transaction value per customer. First, calculate the **total transaction value for each customer** by joining the **Sales and Products tables** (using the product price and sales quantity), then group the results by customer ID.

Step 2 : Identify Customers with the Highest Transaction Value. Next, use the previous **Common Table Expression (CTE)** to find the **CustomerID with the maximum TotalSpent**. Since there may be more than one customer with the same highest amount, this query identifies all such IDs.

Step 3 : Retrieve all transactions of top-spending customers and calculate cumulative amounts over time.

Finally, the main query will retrieve all original transactions for the customer identified above and compute their cumulative amount, ordered by transaction date (Salesdate). This query will return a list of each transaction made by the top-spending customer(s), along with the cumulative total that increases over time.

Row	salesid	customerid	salesdate	transaction_amount	cumulative_amount
1	6261781	94800	2018-02-05 16:35:29.860000 UTC	1.0776000000000001	125916.9072
2	1690999	94800	2018-04-05 06:59:07.100000 UTC	31.0944	30499.212
3	1776321	94800	2018-02-27 00:15:18.240000 UTC	57.336	30556.548
4	5387467	94800	2018-05-09 04:49:01.960000 UTC	57.336	112189.5648
5	1249999	94800	2018-03-31 15:32:32.660000 UTC	70.3848	21273.7296
6	2223238	94800	2018-04-08 07:26:30.240000 UTC	70.3848	38549.7504
7	5296002	94800	2018-02-27 22:23:52.060000 UTC	146.3424	108292.56
8	5480178	94800	2018-02-12 23:11:18.700000 UTC	163.9968	115998.4032
9	2026163	94800	2018-03-07 16:45:57.070000 UTC	182.44559999999998	34205.016
10	6489884	94800	2018-03-21 09:28:25.840000 UTC	212.2296	129460.0008
11	1925405	94800	2018-03-18 20:13:15.130000 UTC	254.31359999999998	33768.2568
12	1949208	94800	2018-03-25 19:14:44.030000 UTC	254.31359999999998	34022.5704
13	4459652	94800	2018-01-04 15:06:45.760000 UTC	342.4512000000003	91065.3312
14	5816313	94800	2018-01-05 01:40:34.040000 UTC	377.3592	121071.9072

Row	salesid	customerid	salesdate	transaction_amount	cumulative_amount
101	3189748	94800	2018-04-08 10:58:54.280000 UTC	2357.9304	61458.4848
102	3124146	94800	2018-01-01 02:19:11.270000 UTC	2366.3472	59100.5544
103	903802	94800	2018-03-17 09:07:18.750000 UTC	2397.012	18178.7616

From the SQL query results, we can see that **the user with the highest transaction value is the customer with ID 94800**. The output provides a detailed transaction history specific to this customer. **All transactions occurred within the year 2018 on specific dates**. The cumulative amount reflects the accumulation of the customer's spending over time, offering a detailed view of the aggregate spending pattern of customer ID 94800.

ACTIONABLE BUSINESS STRATEGIES *using OBIPR*

Objective	Background	Insight	Prioritization	Recommendation
<ul style="list-style-type: none"> Maximize revenue growth by optimizing sales volume and leveraging a stable customer base and reduce dependency on high-volume categories (such as Confections). Maximize customer lifetime value (CLV) through loyalty and increase transaction frequency and value. Optimize growth by addressing revenue anomalies and leverage niche categories (such as Snails) for premium margins. 	<ul style="list-style-type: none"> Strong correlation between total_units_sold and revenue_after_discount. Confections dominate both units sold and revenue Stable customer base (97,300–98,700) and near-perfect correlation between total customers and repeat customers. Customer counts are nearly the same, but revenues differ significantly. Confections outperform Shellfish by more than 250 million. Snails generate 372 million, surpassing Beverages, Produce, and Dairy. 	<ul style="list-style-type: none"> Current growth is volume driven rather than transaction value driven. Stable customer base give opportunity to increase average transaction value (ATV) and cross sell and strong loyalty opportunity to increase transaction value. Growth lies more in ATV and frequency rather than new customer acquisition. Revenue is determined by ATV and purchase frequency, not by customer count. Like, Snails is a niche market with high prices or strong customer loyalty. 	<ul style="list-style-type: none"> Diversify Revenue Drivers, develop strategies to boost ATV in lower-volume categories (e.g., premium pricing, bundling). Cross-Selling Campaigns, use stable customer base to promote complementary products (e.g., pairing Confections with Beverages). Customer Loyalty Programs, incentivize repeat purchases across categories to balance revenue distribution. Category Margin Optimization, identify categories with strong margins (e.g., Grain, Protein) and push targeted promotions. 	<ul style="list-style-type: none"> Revenue differences are driven by transaction value and purchase frequency, not customer count. By optimizing ATV, boosting frequency, and expanding niche categories like Snails, the business can diversify revenue streams, reduce risk, and unlock sustainable growth. The business currently thrives on volume driven revenue with a stable customer base. By shifting focus toward value-driven growth (margins and cross-selling), can balance revenue streams, reduce risk from category dependency, and unlock sustainable long-term growth rather than simply acquiring new customers.

...Thank You...