Report 1: Sales Volume/Category (Low, Moderate, High), Stock Status (Restock Needed, Sufficient), Promotion Suggestion (No Promotion Needed, Consider for promotion), Sales Performance (Above Average, Below Average, No Sales).

Benefits & Business Uses (Value): This table provides valuable insights into each product's monthly sales performance and inventory status across stores, enabling businesses to make data-driven decisions:

- Sales Insights: Identifies top-performing and low-performing products by month, allowing targeted sales strategies.
- Inventory Management: Highlights inventory needs, suggesting restocking for products with low stock relative to demand, helping prevent stockouts and overstock situations.
- Promotion Strategy: Flags products for potential promotions if average sales are low, which can help boost sales for underperforming items.
- Performance Benchmarking: Categorizes products as "Above Average" or "Below Average" based on historical sales, allowing businesses to monitor and respond to trends.
- Sales Categorization: Classifies months into high, moderate, or low sales, assisting in forecasting and seasonal planning, and helping managers align resources with demand patterns.

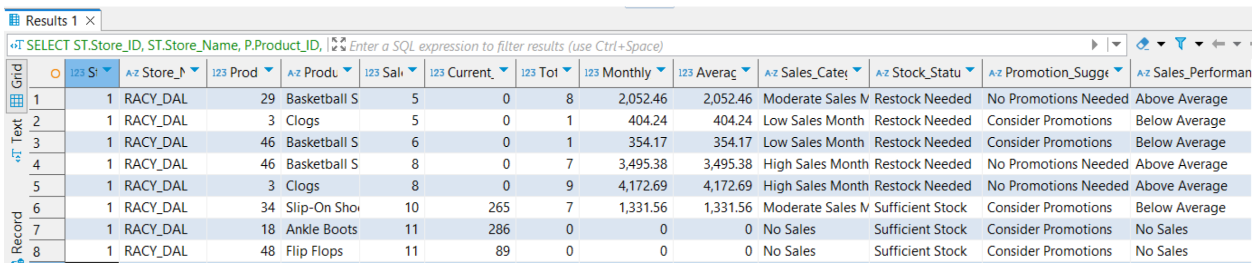
This table aids businesses in aligning inventory and sales strategies with demand, maximizing product performance, and enhancing customer satisfaction by maintaining optimal stock levels.

Assumptions:

- We are assuming that our stores have similar numbers for employees, store selection, and store size with only the sales variable changing across stores.

SQL Query:

```
CREATE VIEW PView_SalesPerformance AS
SELECT ST.Store_ID, ST.Store_Name, P.Product_ID, P.Product_Name,
    EXTRACT(MONTH FROM S.Sale_Date) AS Sale_Month,
    COALESCE(SI.Stock_Quantity, 0) AS Current_Stock,
    SUM(S.Qyantity) AS Total_Sales,
    SUM(S.Total_Amount) AS Monthly_Total_Sales,
    ROUND(AVG(S.Total_Amount), 2) AS Average_Sale_Amount,
    CASE
        WHEN ROUND(SUM(S.Total_Amount), 2) > 3000 THEN 'High Sales Month'
        WHEN ROUND(SUM(S.Total_Amount), 2) BETWEEN 1000 AND 3000 THEN 'Moderate Sales
Month'
        ELSE 'Low Sales Month'
    END AS Sales_Category,
    CASE
        WHEN COALESCE(SI.Stock_Quantity, 0) < (AVG(S.Qyantity) * 2) THEN 'Restock
Needed'
        ELSE 'Sufficient Stock'
    END AS Stock_Status,
    CASE
        WHEN ROUND(AVG(S.Total_Amount), 2) < 2000 THEN 'Consider Promotions'
        ELSE 'No Promotions Needed'
    END AS Promotion_Suggestion,
    CASE
        WHEN SUM(S.Total_Amount) > (
            SELECT AVG(Monthly_Sales)
            FROM (
                SELECT Product_ID, SUM(Total_Amount) AS Monthly_Sales
                FROM P_Sales
                GROUP BY Product_ID, EXTRACT(MONTH FROM Sale_Date)
            ) AS Avg_Monthly_Sales
            WHERE Avg_Monthly_Sales.Product_ID = P.Product_ID
        ) THEN 'Above Average'
        ELSE 'Below Average'
    END AS Sales_Performance
FROM P_Sales S
JOIN P_Product P ON S.Product_ID = P.Product_ID
JOIN P_Store ST ON S.Store_ID = ST.Store_ID
LEFT JOIN P_Store_Inventory SI ON ST.Store_ID = SI.Store_ID AND P.Product_ID =
SI.Product_ID
GROUP BY ST.Store_ID, ST.Store_Name, P.Product_ID, P.Product_Name, Sale_Month,
SI.Stock_Quantity
UNION
-- Part 2: Products without Sales
SELECT ST.Store_ID, ST.Store_Name, P.Product_ID, P.Product_Name,
    EXTRACT(MONTH FROM CURRENT_DATE) AS Sale_Month,
    COALESCE(SI.Stock_Quantity, 0) AS Current_Stock,
    0 AS Total_Sales,
    0 AS Monthly_Total_Sales,
    0 AS Average_Sale_Amount,
    'No Sales' AS Sales_Category,
    CASE
        WHEN COALESCE(SI.Stock_Quantity, 0) < 10 THEN 'Restock Needed'
        ELSE 'Sufficient Stock'
    END AS Stock_Status,
    'Consider Promotions' AS Promotion_Suggestion,
    'No Sales' AS Sales_Performance
FROM P_Product P
JOIN P_Store_Inventory SI ON P.Product_ID = SI.Product_ID
JOIN P_Store ST ON SI.Store_ID = ST.Store_ID
WHERE P.Product_ID NOT IN (
        SELECT DISTINCT Product_ID
        FROM P_Sales
        WHERE EXTRACT(MONTH FROM Sale_Date) = EXTRACT(MONTH FROM CURRENT_DATE)
    )
ORDER BY Store_ID, Sale_Month, Product_Name;
```


Results 1

SELECT ST.Store_ID, ST.Store_Name, P.Product_ID, | Enter a SQL expression to filter results (use Ctrl+Space)

	St	Store_N	Prod	Produ	Sal	Current_	Tot	Monthly	Averag	Sales_Cate	Stock_Statu	Promotion_Sugge	Sales_Performan
1	1	RACY_DAL	29	Basketball S	5	0	8	2,052.46	2,052.46	Moderate Sales M	Restock Needed	No Promotions Needed	Above Average
2	1	RACY_DAL	3	Clogs	5	0	1	404.24	404.24	Low Sales Month	Restock Needed	Consider Promotions	Below Average
3	1	RACY_DAL	46	Basketball S	6	0	1	354.17	354.17	Low Sales Month	Restock Needed	Consider Promotions	Below Average
4	1	RACY_DAL	46	Basketball S	8	0	7	3,495.38	3,495.38	High Sales Month	Restock Needed	No Promotions Needed	Above Average
5	1	RACY_DAL	3	Clogs	8	0	9	4,172.69	4,172.69	High Sales Month	Restock Needed	No Promotions Needed	Above Average
6	1	RACY_DAL	34	Slip-On Sho	10	265	7	1,331.56	1,331.56	Moderate Sales M	Sufficient Stock	Consider Promotions	Below Average
7	1	RACY_DAL	18	Ankle Boots	11	286	0	0	0	No Sales	Sufficient Stock	Consider Promotions	No Sales
8	1	RACY_DAL	48	Flip Flops	11	89	0	0	0	No Sales	Sufficient Stock	Consider Promotions	No Sales

Report 2: Calculating the Total Refund amount and looking at the shipment type.

Benefits & Business Uses (Value):

- Consolidation of Shipment Data
  - Value: By creating the view V_Shipment_Details, the query consolidates data from P_Domestic and P_International tables into a unified view. This allows easy access to shipment type information (domestic or international) without querying multiple tables.
  - Use: Simplifies reporting and analysis for shipment details, ensuring consistency across teams that need shipment-related insights.
- Enhanced Return Management
  - Value: The query provides a detailed breakdown of return information, including shipment type and refund type (e.g., full, partial). This helps businesses identify patterns in returns, such as high return rates for specific shipment types or reasons.
  - Use: Can be used by operations and customer service teams to improve return policies, optimize logistics, and reduce operational costs.
- Customer Experience Insights
  - Value: By analyzing the return reasons and refund amounts, businesses can understand customer dissatisfaction points, whether it's due to product quality, shipment issues, or other factors.
  - Use: Helps improve customer satisfaction by addressing common return reasons and enhancing product or shipping standards.
- Financial Analysis
  - Value: The calculated Refund_Amount provides insight into financial implications of returns (e.g., total refund costs by shipment type or reason).
  - Use: Enables financial teams to forecast refund liabilities and measure the cost impact of return policies on overall profitability.
- Operational Optimization
  - Value: By identifying shipment types tied to specific return patterns, logistics and supply chain teams can optimize shipment strategies to minimize returns.
  - Use: Drives operational efficiency by aligning shipping methods or inventory management practices with customer needs.

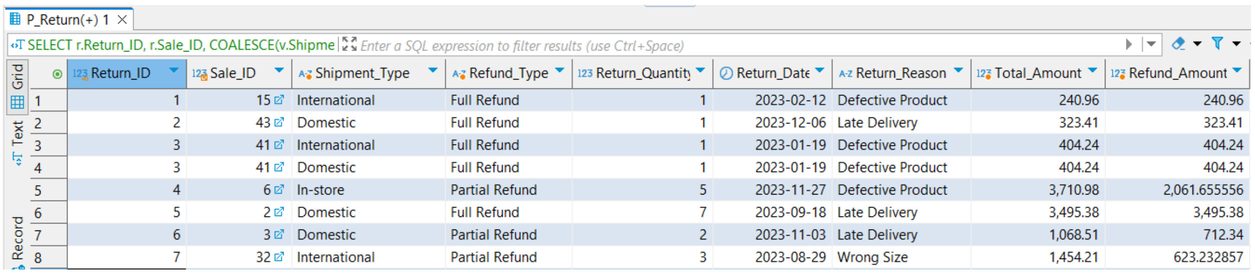
Assumptions:

- A full refund occurs when Return_Quantity equals Quantity.
- A partial refund occurs when Return_Quantity is less than Quantity.
- Any other scenario is considered 'Unknown'.
- Return_Quantity in the P_Return table is always a valid, non-negative number that does not exceed the original Quantity sold.

SQL Query:

```sql
-- Drop the view if it exists
DROP VIEW IF EXISTS P_Shipment_Details;
-- Create the view V_Shipment_Details
CREATE VIEW P_Shipment_Details AS
SELECT
    Shipment_ID,
    'Domestic' AS Shipment_Type
FROM
    P_Domestic
```

```
UNION
SELECT
    Shipment_ID,
    'International' AS Shipment_Type
FROM
    P_International;
-- Main query
SELECT
    r.Return_ID,
    r.Sale_ID,
    COALESCE(v.Shipment_Type, 'In-store') AS Shipment_Type,
    CASE
        WHEN r.Return_Quantity = s.Quantity THEN 'Full Refund'
        WHEN r.Return_Quantity < s.Quantity THEN 'Partial Refund'
        ELSE 'Unknown'
    END AS Refund_Type,
    r.Return_Quantity,
    r.Return_Date,
    r.Return_Reason,
    s.Total_Amount,
    (s.Total_Amount / s.Quantity) * r.Return_Quantity AS Refund_Amount
FROM P_Return r
LEFT JOIN P_Shipment_Details v ON r.Return_ID = v.Shipment_ID
JOIN P_Sales s ON r.Sale_ID = s.Sale_ID
ORDER BY r.Return_ID;
```



P_Return(+) 1

SELECT r.Return_ID, r.Sale_ID, COALESCE(v.Shipme

	Return_ID	Sale_ID	Shipment_Type	Refund_Type	Return_Quantity	Return_Date	Return_Reason	Total_Amount	Refund_Amount
1	1	15	International	Full Refund	1	2023-02-12	Defective Product	240.96	240.96
2	2	43	Domestic	Full Refund	1	2023-12-06	Late Delivery	323.41	323.41
3	3	41	International	Full Refund	1	2023-01-19	Defective Product	404.24	404.24
4	3	41	Domestic	Full Refund	1	2023-01-19	Defective Product	404.24	404.24
5	4	6	In-store	Partial Refund	5	2023-11-27	Defective Product	3,710.98	2,061.655556
6	5	2	Domestic	Full Refund	7	2023-09-18	Late Delivery	3,495.38	3,495.38
7	6	3	Domestic	Partial Refund	2	2023-11-03	Late Delivery	1,068.51	712.34
8	7	32	International	Partial Refund	3	2023-08-29	Wrong Size	1,454.21	623.232857

Report 3: Report 2 from submission 5a using window function.

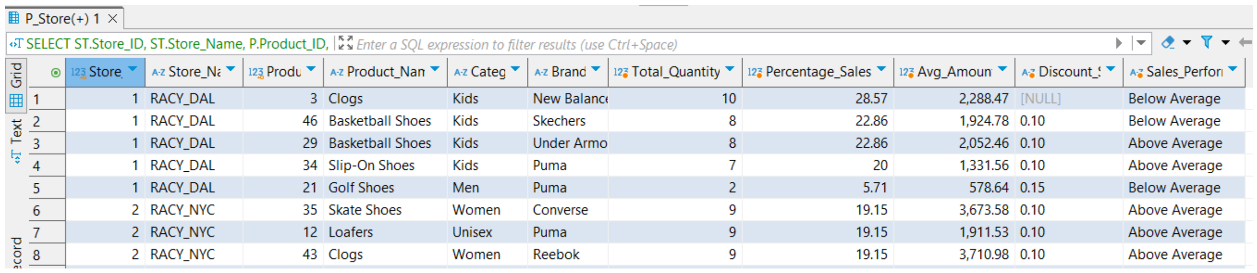
Benefits & Business Uses (Value): This query provides comprehensive insights into product performance at the store level, empowering store managers and decision-makers to identify high and low-performing products. By understanding each product's contribution to overall sales, stores can make informed decisions about inventory management, promotions, and discounts. The discount suggestion metric is especially valuable for optimizing pricing strategies to boost sales of slower-moving items, enhancing revenue and inventory turnover. Additionally, tracking average spending per product helps tailor marketing strategies and refine product placements to match customer preferences, resulting in improved customer satisfaction and retention, as well as maximizing profitability per square foot of retail space.

Assumptions:
We are binning the quantities sold as follows: 0 to 3, It will give a 15% promotional discount. 4 to 9, it will give a 10% discount. More than 9, it won't give any discount.

SQL Query:
```
SELECT
    ST.Store_ID,
    ST.Store_Name,
    P.Product_ID,
    P.Product_Name,
    P.Category,
    P.Brand,
    SUM(S.Quantity) AS Total_Quantity_Sold, -- Total quantity sold per product per
store
    ROUND((SUM(S.Quantity) * 100.0 / SUM(SUM(S.Quantity)) OVER (PARTITION BY
ST.Store_ID)), 2) AS Percentage_Sales, -- Percentage of total sales by each product
in the store
    ROUND(AVG(S.Total_Amount), 2) AS Avg_Amount, -- Average amount spent on each
product
    CASE
        WHEN SUM(S.Quantity) BETWEEN 4 AND 9 THEN '0.10'
        WHEN SUM(S.Quantity) <= 3 THEN '0.15'
        ELSE NULL
    END AS Discount_Suggestion, -- Discount based on the total quantity sold
    CASE
        WHEN SUM(S.Quantity) > AVG(SUM(S.Quantity)) OVER (PARTITION BY P.Product_ID)
THEN 'Above Average'
        ELSE 'Below Average'
    END AS Sales_Performance -- Categorize as Above or Below Average based on total
quantity sold
FROM
    P_Sales S
JOIN
    P_Product P ON S.Product_ID = P.Product_ID
JOIN
    P_Store ST ON S.Store_ID = ST.Store_ID
GROUP BY
    ST.Store_ID, ST.Store_Name, P.Product_ID, P.Product_Name, P.Category, P.Brand
HAVING
    Total_Quantity_Sold > 0
```

```sql
ORDER BY
    ST.Store_ID, Total_Quantity_Sold DESC;
```

New Screenshot:

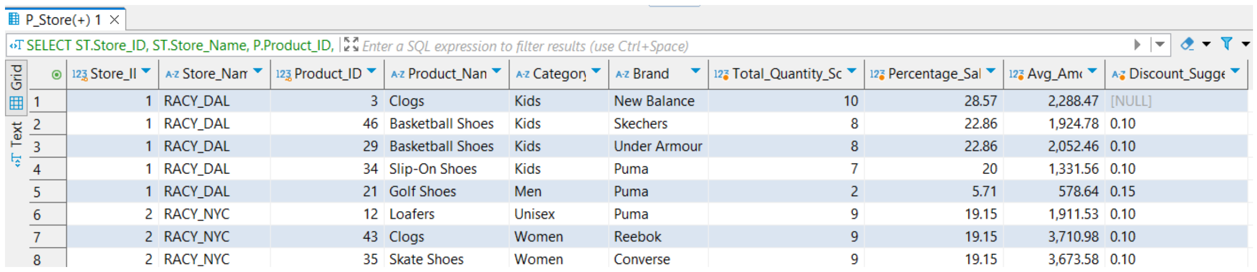


P_Store(+) 1

SELECT ST.Store_ID, ST.Store_Name, P.Product_ID, | Enter a SQL expression to filter results (use Ctrl+Space)

	Store_	Store_Na	Produ	Product_Nan	Categ	Brand	Total_Quantity	Percentage_Sales	Avg_Amoun	Discount_!	Sales_Perfori
1	1	RACY_DAL	3	Clogs	Kids	New Balance	10	28.57	2,288.47	[NULL]	Below Average
2	1	RACY_DAL	46	Basketball Shoes	Kids	Skechers	8	22.86	1,924.78	0.10	Below Average
3	1	RACY_DAL	29	Basketball Shoes	Kids	Under Armo	8	22.86	2,052.46	0.10	Above Average
4	1	RACY_DAL	34	Slip-On Shoes	Kids	Puma	7	20	1,331.56	0.10	Above Average
5	1	RACY_DAL	21	Golf Shoes	Men	Puma	2	5.71	578.64	0.15	Below Average
6	2	RACY_NYC	35	Skate Shoes	Women	Converse	9	19.15	3,673.58	0.10	Above Average
7	2	RACY_NYC	12	Loafers	Unisex	Puma	9	19.15	1,911.53	0.10	Above Average
8	2	RACY_NYC	43	Clogs	Women	Reebok	9	19.15	3,710.98	0.10	Above Average

Old Screenshot:



P_Store(+) 1

SELECT ST.Store_ID, ST.Store_Name, P.Product_ID, | Enter a SQL expression to filter results (use Ctrl+Space)

	Store_II	Store_Nam	Product_ID	Product_Nan	Category	Brand	Total_Quantity_Sc	Percentage_Sal	Avg_Amc	Discount_Sugge
1	1	RACY_DAL	3	Clogs	Kids	New Balance	10	28.57	2,288.47	[NULL]
2	1	RACY_DAL	46	Basketball Shoes	Kids	Skechers	8	22.86	1,924.78	0.10
3	1	RACY_DAL	29	Basketball Shoes	Kids	Under Armour	8	22.86	2,052.46	0.10
4	1	RACY_DAL	34	Slip-On Shoes	Kids	Puma	7	20	1,331.56	0.10
5	1	RACY_DAL	21	Golf Shoes	Men	Puma	2	5.71	578.64	0.15
6	2	RACY_NYC	12	Loafers	Unisex	Puma	9	19.15	1,911.53	0.10
7	2	RACY_NYC	43	Clogs	Women	Reebok	9	19.15	3,710.98	0.10
8	2	RACY_NYC	35	Skate Shoes	Women	Converse	9	19.15	3,673.58	0.10