RetailCo Inventory Analytics – SQL Case Study

Overview

This document presents SQL queries and insights for the RetailCo Inventory Analytics project.

It demonstrates business-focused analysis using inventory, sales, supplier, and product data.

1. Total Sales Revenue by Store

```
SELECT s.StoreName, SUM(sa.QuantitySold * p.SellingPrice) AS TotalRevenue
FROM Sales sa
JOIN Stores s ON sa.StoreID = s.StoreID
JOIN Products p ON sa.ProductID = p.ProductID
GROUP BY s.StoreName
ORDER BY TotalRevenue DESC;
```

Insight: Identifies highest-performing store locations.

2. Monthly Sales Trend

```
SELECT DATE_TRUNC('month', SaleDate) AS Month,
SUM(sa.QuantitySold * p.SellingPrice) AS MonthlyRevenue
FROM Sales sa
JOIN Products p ON sa.ProductID = p.ProductID
GROUP BY Month
ORDER BY Month;
```

Insight: Shows seasonal patterns and growth trends.

3. Top 10 Best-Selling Products

```
SELECT p.ProductName, SUM(sa.QuantitySold) AS TotalUnitsSold
FROM Sales sa
```

```
JOIN Products p ON sa.ProductID = p.ProductID
GROUP BY p.ProductName
ORDER BY TotalUnitsSold DESC
LIMIT 10;
```

4. Slow-Moving / Dead Stock

```
SELECT p.ProductName, i.StoreID, i.CurrentStock
FROM Inventory i
JOIN Products p ON i.ProductID = p.ProductID
WHERE i.ProductID NOT IN (
SELECT DISTINCT ProductID FROM Sales
WHERE SaleDate >= CURRENT_DATE - INTERVAL '60 days');
```

Insight: Helps reduce carrying cost and optimize stock levels.

5. Inventory Turnover per Product

```
SELECT p.ProductName, SUM(sa.QuantitySold) AS UnitsSold,
AVG(i.CurrentStock) AS AvgInventory,
(SUM(sa.QuantitySold)/NULLIF(AVG(i.CurrentStock),0)) AS InventoryTurnover
FROM Sales sa
JOIN Inventory i ON sa.ProductID = i.ProductID AND sa.StoreID = i.StoreID
JOIN Products p ON sa.ProductID = p.ProductID
GROUP BY p.ProductName
ORDER BY InventoryTurnover DESC;
```

6. Products Below Reorder Point

```
SELECT p.ProductName, i.StoreID, i.CurrentStock, i.ReorderPoint
FROM Inventory i
```

```
JOIN Products p ON i.ProductID = p.ProductID
WHERE i.CurrentStock < i.ReorderPoint;
```

Insight: Critical for preventing stockouts.

7. Revenue Contribution by Category

```
SELECT p.Category,
SUM(sa.QuantitySold * p.SellingPrice) AS CategoryRevenue
FROM Sales sa
JOIN Products p ON sa.ProductID = p.ProductID
GROUP BY p.Category;
```

8. Lead Time by Supplier

```
SELECT SupplierName, AVG(LeadTimeDays) AS AvgLeadTime
FROM Suppliers
GROUP BY SupplierName;
```

9. Purchase Orders vs Sales

```
SELECT p.ProductName, SUM(po.QuantityOrdered) AS TotalOrdered,
SUM(sa.QuantitySold) AS TotalSold,
SUM(po.QuantityOrdered) - SUM(sa.QuantitySold) AS StockBalance
FROM PurchaseOrders po
JOIN Sales sa ON po.ProductID = sa.ProductID
JOIN Products p ON p.ProductID = sa.ProductID
GROUP BY p.ProductName;
```

10. Store-Level Product Ranking

```
SELECT s.StoreName, p.ProductName, SUM(sa.QuantitySold) AS UnitsSold,
RANK() OVER (PARTITION BY s.StoreName ORDER BY SUM(sa.QuantitySold) DESC) AS
ProductRank
```

```
FROM Sales sa

JOIN Stores s ON sa.StoreID = s.StoreID

JOIN Products p ON sa.ProductID = p.ProductID

GROUP BY s.StoreName, p.ProductName;
```

Conclusion

These SQL queries form a complete data-driven view of inventory, sales, and supplier efficiency for RetailCo.

They can be showcased directly in a portfolio or GitHub repository.