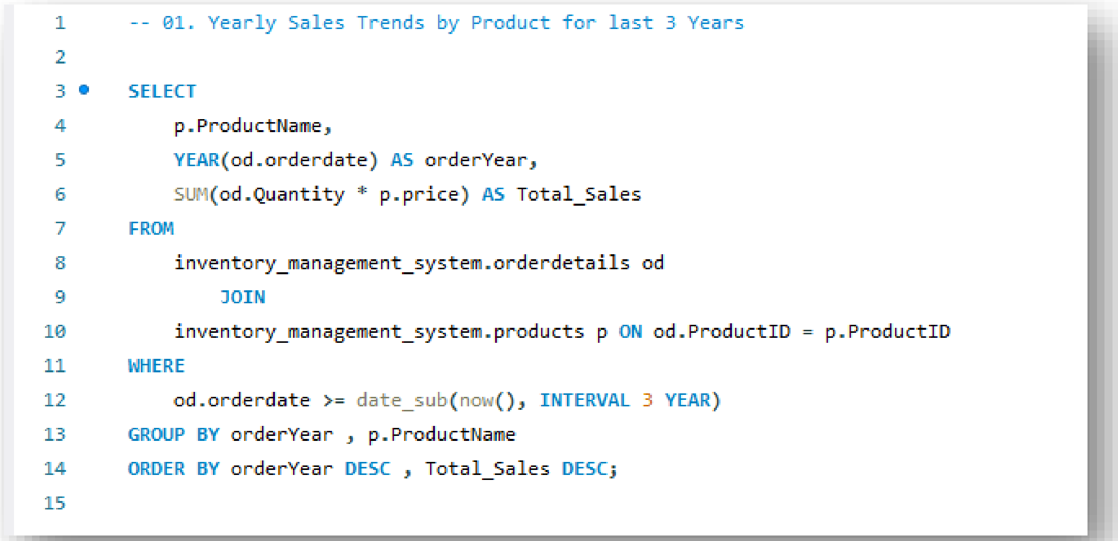
Inventory Management System

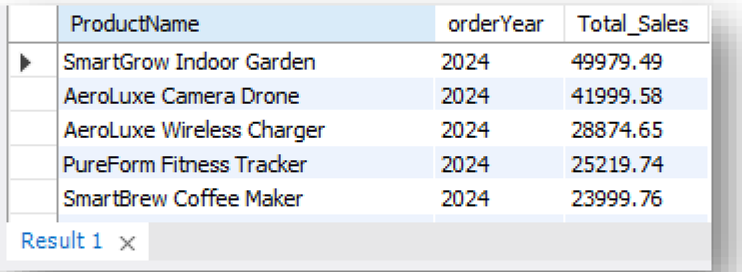
1. Yearly Sales Trends by Product for last 3 Years

MySQL Query



```sql
-- 01. Yearly Sales Trends by Product for last 3 Years

SELECT
    p.ProductName,
    YEAR(od.orderdate) AS orderYear,
    SUM(od.Quantity * p.price) AS Total_Sales
FROM
    inventory_management_system.orderdetails od
        JOIN
    inventory_management_system.products p ON od.ProductID = p.ProductID
WHERE
    od.orderdate >= date_sub(now(), INTERVAL 3 YEAR)
GROUP BY orderYear , p.ProductName
ORDER BY orderYear DESC , Total_Sales DESC;
```

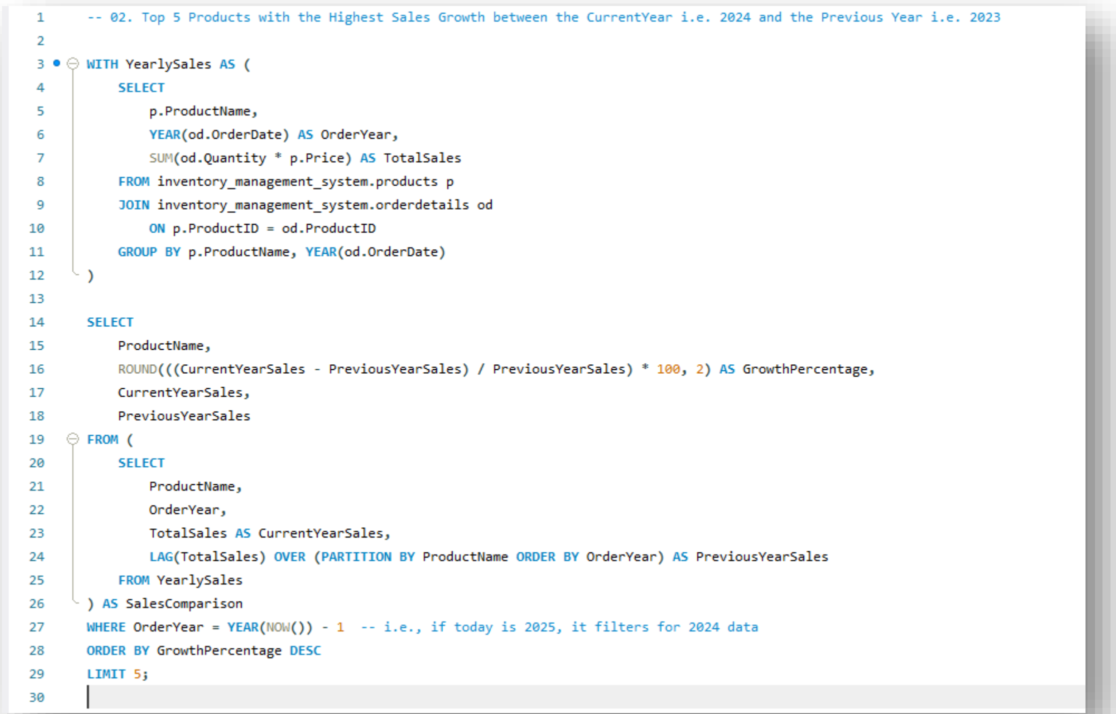
Its Result



ProductName	orderYear	Total_Sales
SmartGrow Indoor Garden	2024	49979.49
AeroLuxe Camera Drone	2024	41999.58
AeroLuxe Wireless Charger	2024	28874.65
PureForm Fitness Tracker	2024	25219.74
SmartBrew Coffee Maker	2024	23999.76

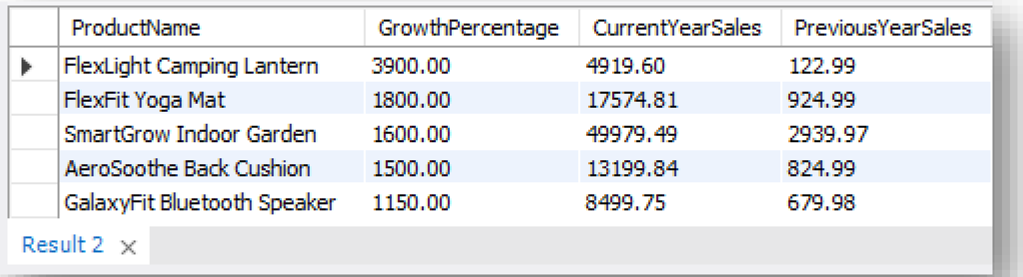
02. Top 5 Products with the Highest Sales Growth between the Current Year i.e., 2024 and the Previous Year i.e., 2023

MySQL Query



```sql
-- 02. Top 5 Products with the Highest Sales Growth between the CurrentYear i.e. 2024 and the Previous Year i.e. 2023

WITH YearlySales AS (
    SELECT
        p.ProductName,
        YEAR(od.OrderDate) AS OrderYear,
        SUM(od.Quantity * p.Price) AS TotalSales
    FROM inventory_management_system.products p
    JOIN inventory_management_system.orderdetails od
        ON p.ProductID = od.ProductID
    GROUP BY p.ProductName, YEAR(od.OrderDate)
)

SELECT
    ProductName,
    ROUND(((CurrentYearSales - PreviousYearSales) / PreviousYearSales) * 100, 2) AS GrowthPercentage,
    CurrentYearSales,
    PreviousYearSales
FROM (
    SELECT
        ProductName,
        OrderYear,
        TotalSales AS CurrentYearSales,
        LAG(TotalSales) OVER (PARTITION BY ProductName ORDER BY OrderYear) AS PreviousYearSales
    FROM YearlySales
) AS SalesComparison
WHERE OrderYear = YEAR(NOW()) - 1  -- i.e., if today is 2025, it filters for 2024 data
ORDER BY GrowthPercentage DESC
LIMIT 5;
```

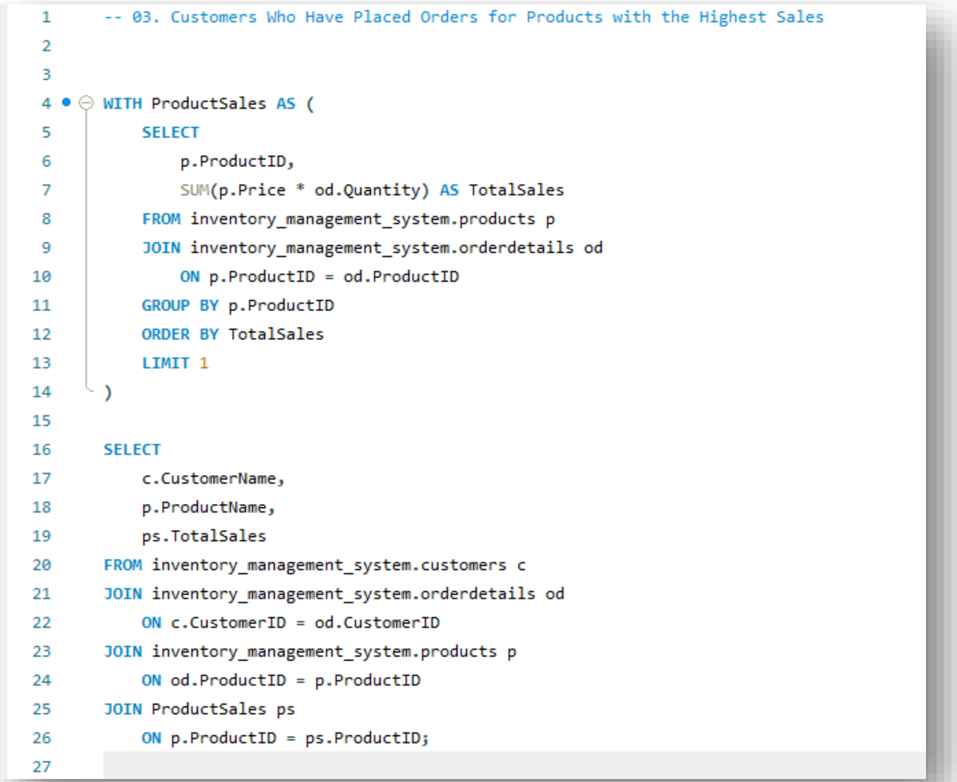
Its Result



ProductName	GrowthPercentage	CurrentYearSales	PreviousYearSales
FlexLight Camping Lantern	3900.00	4919.60	122.99
FlexFit Yoga Mat	1800.00	17574.81	924.99
SmartGrow Indoor Garden	1600.00	49979.49	2939.97
AeroSoothe Back Cushion	1500.00	13199.84	824.99
GalaxyFit Bluetooth Speaker	1150.00	8499.75	679.98

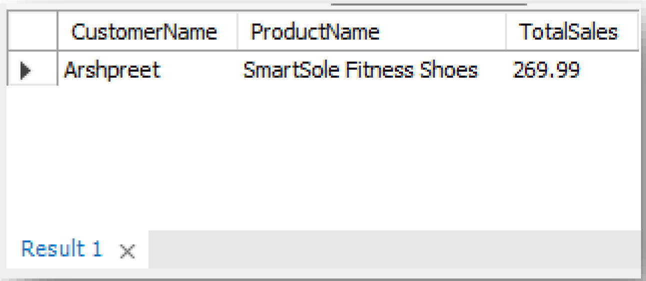
3. Customers Who Have Placed Orders for Products with the Highest Sales

MySQL Query



```sql
-- 03. Customers Who Placed Orders for Products with the Highest Sales


WITH ProductSales AS (
    SELECT
        p.ProductID,
        SUM(p.Price * od.Quantity) AS TotalSales
    FROM inventory_management_system.products p
    JOIN inventory_management_system.orderdetails od
        ON p.ProductID = od.ProductID
    GROUP BY p.ProductID
    ORDER BY TotalSales
    LIMIT 1
)

SELECT
    c.CustomerName,
    p.ProductName,
    ps.TotalSales
FROM inventory_management_system.customers c
JOIN inventory_management_system.orderdetails od
    ON c.CustomerID = od.CustomerID
JOIN inventory_management_system.products p
    ON od.ProductID = p.ProductID
JOIN ProductSales ps
    ON p.ProductID = ps.ProductID;

```

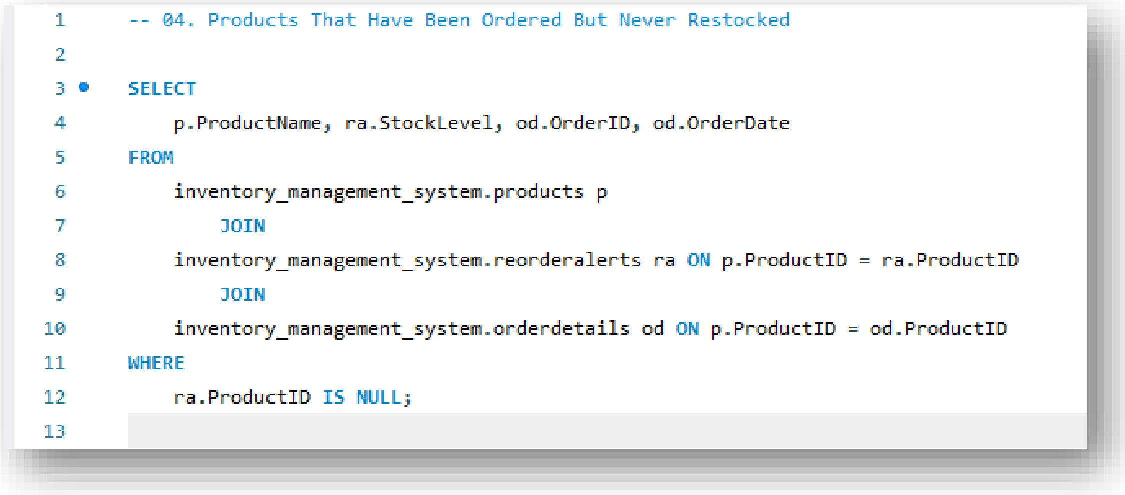
Its Result



CustomerName	ProductName	TotalSales
Arshpreet	SmartSole Fitness Shoes	269.99

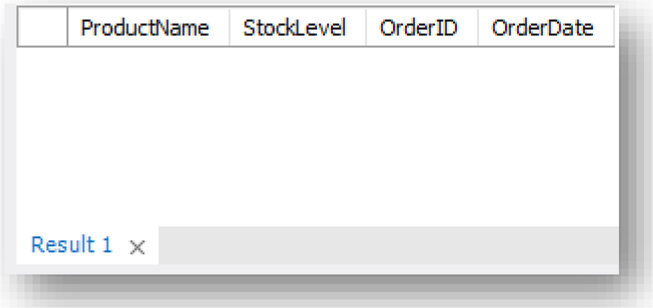
4. Products That Have Been Ordered But Never Restocked

MySQL Query



```
-- 04. Products That Have Been Ordered But Never Restocked

SELECT
    p.ProductName, ra.StockLevel, od.OrderID, od.OrderDate
FROM
    inventory_management_system.products p
        JOIN
    inventory_management_system.reorderalerts ra ON p.ProductID = ra.ProductID
        JOIN
    inventory_management_system.orderdetails od ON p.ProductID = od.ProductID
WHERE
    ra.ProductID IS NULL;
```

Its Result

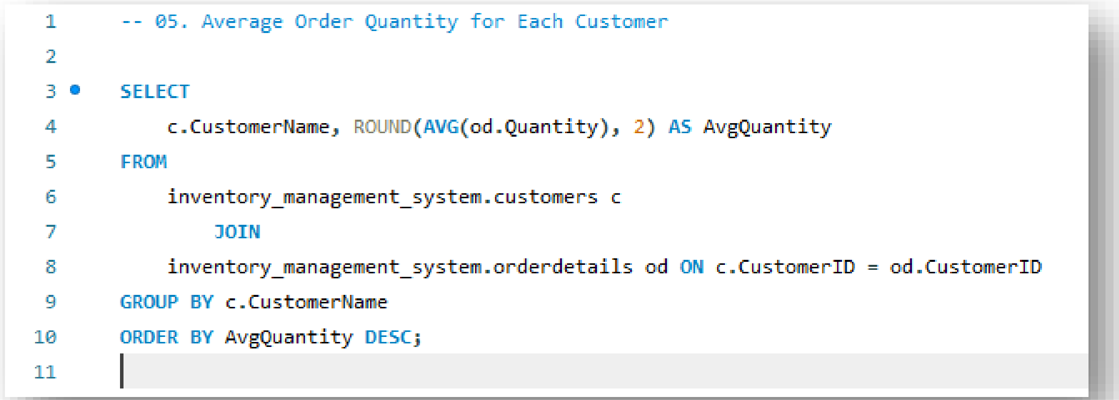


	ProductName	StockLevel	OrderID	OrderDate

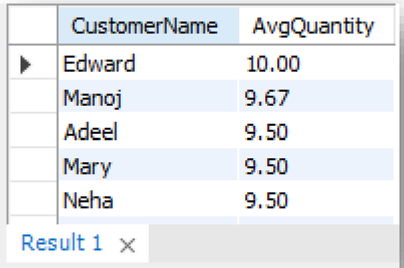
5. Average Order Quantity for Each Customer

MySQL Query

Its Result



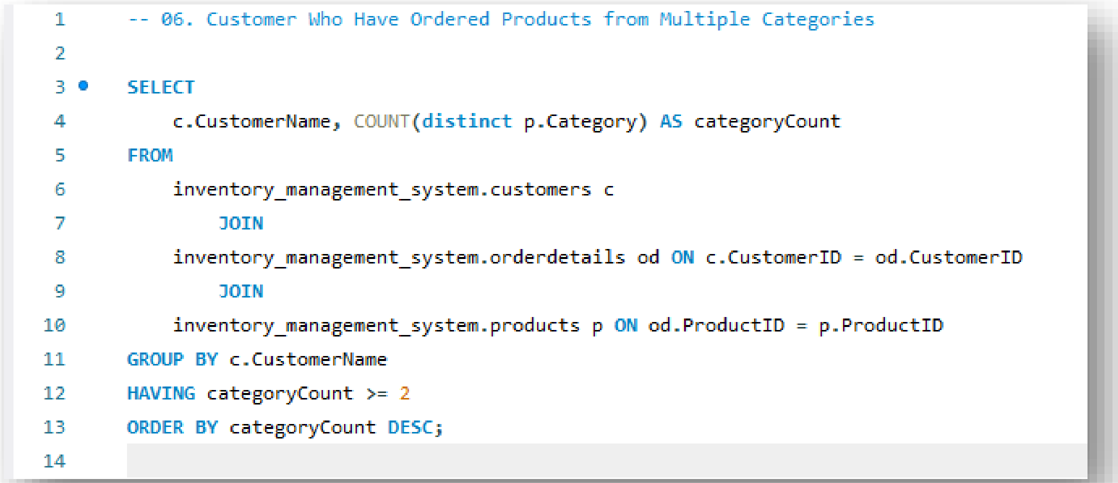
```sql
-- 05. Average Order Quantity for Each Customer

SELECT
    c.CustomerName, ROUND(AVG(od.Quantity), 2) AS AvgQuantity
FROM
    inventory_management_system.customers c
        JOIN
    inventory_management_system.orderdetails od ON c.CustomerID = od.CustomerID
GROUP BY c.CustomerName
ORDER BY AvgQuantity DESC;
```



CustomerName	AvgQuantity
Edward	10.00
Manoj	9.67
Adeel	9.50
Mary	9.50
Neha	9.50

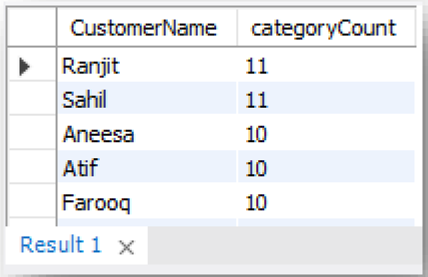
6. Customer Who Have Ordered Products from Multiple Categories

MySQL Query



```sql
-- 06. Customer Who Have Ordered Products from Multiple Categories

SELECT
    c.CustomerName, COUNT(distinct p.Category) AS categoryCount
FROM
    inventory_management_system.customers c
        JOIN
    inventory_management_system.orderdetails od ON c.CustomerID = od.CustomerID
        JOIN
    inventory_management_system.products p ON od.ProductID = p.ProductID
GROUP BY c.CustomerName
HAVING categoryCount >= 2
ORDER BY categoryCount DESC;
```

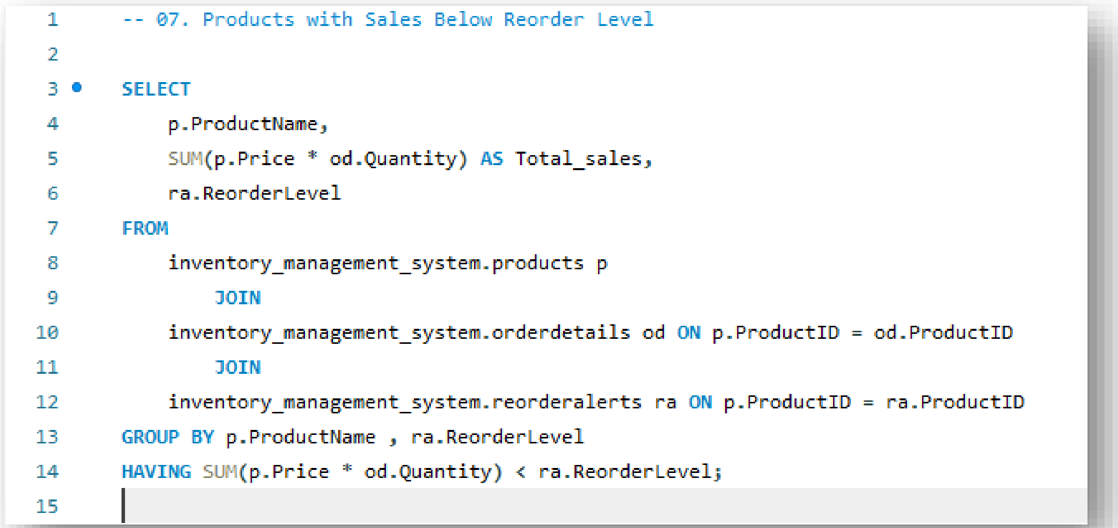
Its Result



CustomerName	categoryCount
Ranjit	11
Sahil	11
Aneesa	10
Atif	10
Farooq	10

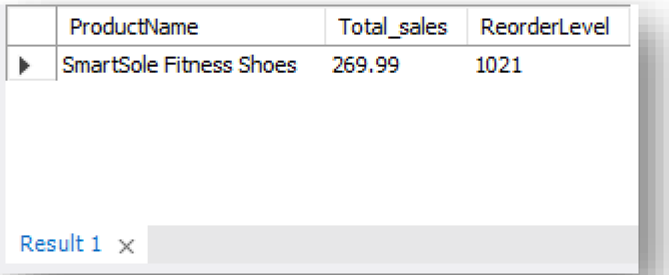
7. Products with Sales Below Reorder Level

MySQL Query



```sql
-- 07. Products with Sales Below Reorder Level

SELECT
    p.ProductName,
    SUM(p.Price * od.Quantity) AS Total_sales,
    ra.ReorderLevel
FROM
    inventory_management_system.products p
        JOIN
    inventory_management_system.orderdetails od ON p.ProductID = od.ProductID
        JOIN
    inventory_management_system.reorderalerts ra ON p.ProductID = ra.ProductID
GROUP BY p.ProductName , ra.ReorderLevel
HAVING SUM(p.Price * od.Quantity) < ra.ReorderLevel;
```

Its Result

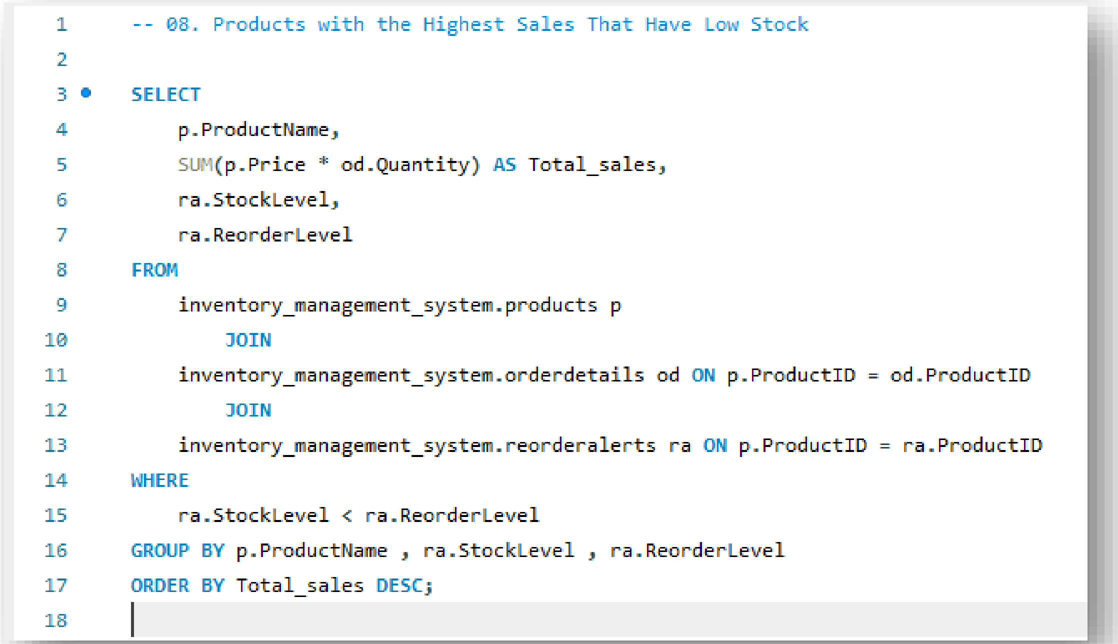


ProductName	Total_sales	ReorderLevel
SmartSole Fitness Shoes	269.99	1021

Result 1

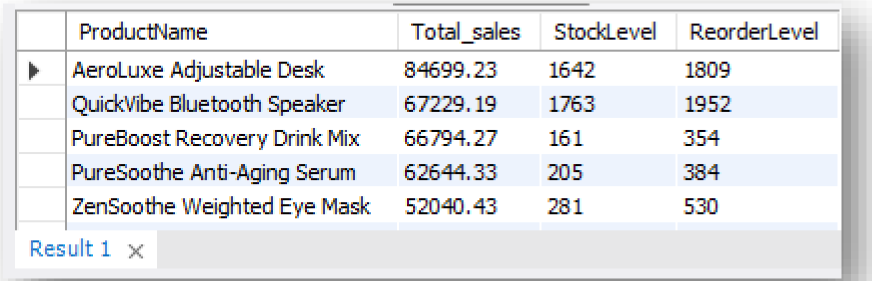
8. Products with the Highest Sales That Have Low Stock

MySQL Query



```sql
-- 08. Products with the Highest Sales That Have Low Stock

SELECT
    p.ProductName,
    SUM(p.Price * od.Quantity) AS Total_sales,
    ra.StockLevel,
    ra.ReorderLevel
FROM
    inventory_management_system.products p
        JOIN
    inventory_management_system.orderdetails od ON p.ProductID = od.ProductID
        JOIN
    inventory_management_system.reorderalerts ra ON p.ProductID = ra.ProductID
WHERE
    ra.StockLevel < ra.ReorderLevel
GROUP BY p.ProductName , ra.StockLevel , ra.ReorderLevel
ORDER BY Total_sales DESC;
```

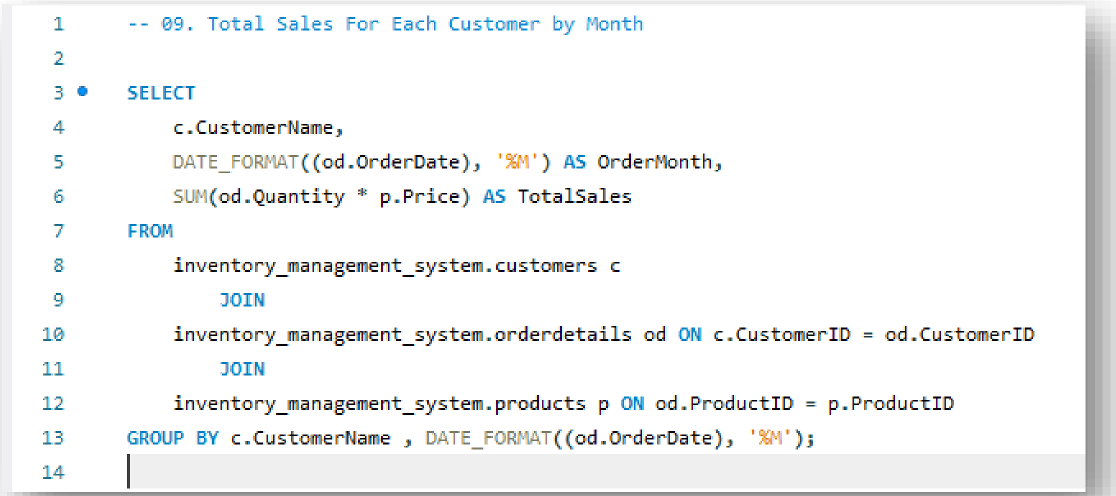
Its Result



ProductName	Total_sales	StockLevel	ReorderLevel
AeroLuxe Adjustable Desk	84699.23	1642	1809
QuickVibe Bluetooth Speaker	67229.19	1763	1952
PureBoost Recovery Drink Mix	66794.27	161	354
PureSoothe Anti-Aging Serum	62644.33	205	384
ZenSoothe Weighted Eye Mask	52040.43	281	530

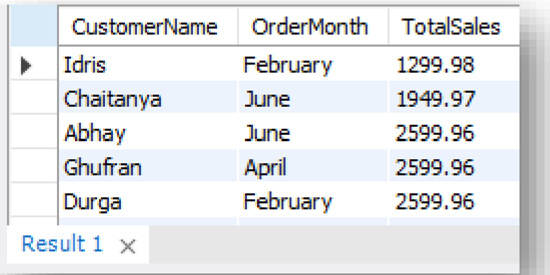
9. Total Sales For Each Customer by Month

MySQL Query



```sql
-- 09. Total Sales For Each Customer by Month

SELECT
    c.CustomerName,
    DATE_FORMAT((od.OrderDate), '%M') AS OrderMonth,
    SUM(od.Quantity * p.Price) AS TotalSales
FROM
    inventory_management_system.customers c
        JOIN
    inventory_management_system.orderdetails od ON c.CustomerID = od.CustomerID
        JOIN
    inventory_management_system.products p ON od.ProductID = p.ProductID
GROUP BY c.CustomerName , DATE_FORMAT((od.OrderDate), '%M');
```

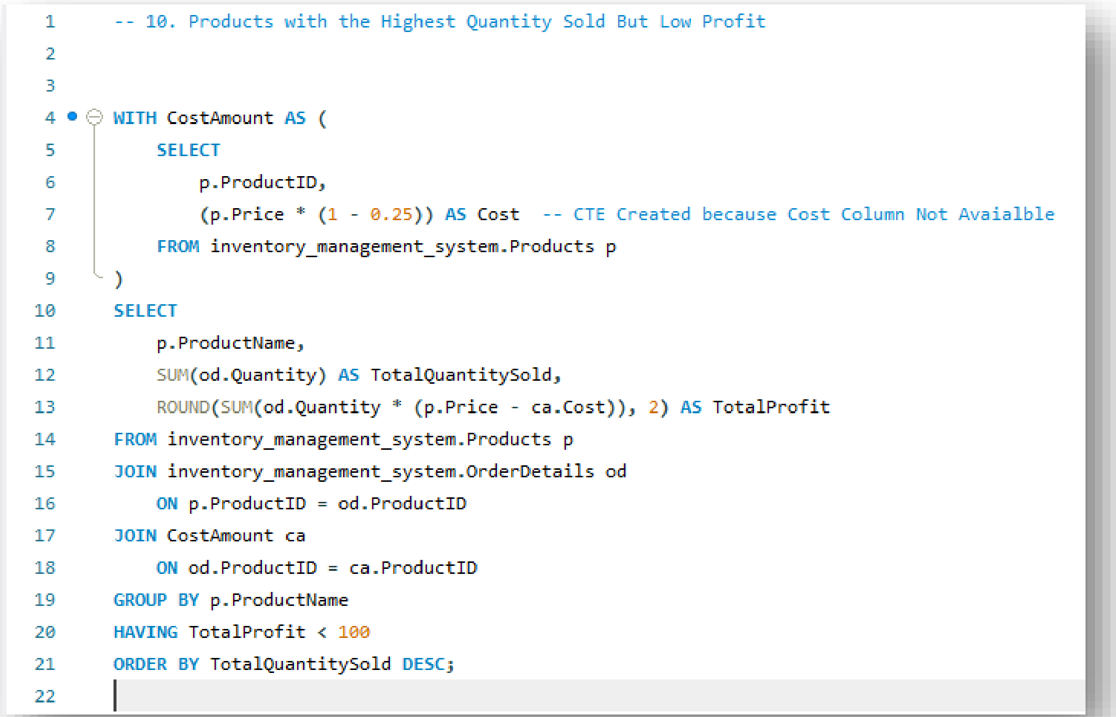
Its Result



CustomerName	OrderMonth	TotalSales
Idris	February	1299.98
Chaitanya	June	1949.97
Abhay	June	2599.96
Ghufran	April	2599.96
Durga	February	2599.96

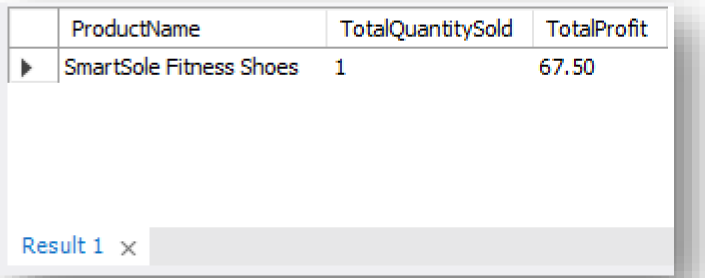
10. Products with the Highest Quantity Sold But Low Profit

MySQL Query



```sql
-- 10. Products with the Highest Quantity Sold But Low Profit


WITH CostAmount AS (
    SELECT
        p.ProductID,
        (p.Price * (1 - 0.25)) AS Cost  -- CTE Created because Cost Column Not Avaialble
    FROM inventory_management_system.Products p
)
SELECT
    p.ProductName,
    SUM(od.Quantity) AS TotalQuantitySold,
    ROUND(SUM(od.Quantity * (p.Price - ca.Cost)), 2) AS TotalProfit
FROM inventory_management_system.Products p
JOIN inventory_management_system.OrderDetails od
    ON p.ProductID = od.ProductID
JOIN CostAmount ca
    ON od.ProductID = ca.ProductID
GROUP BY p.ProductName
HAVING TotalProfit < 100
ORDER BY TotalQuantitySold DESC;
```

Its Result

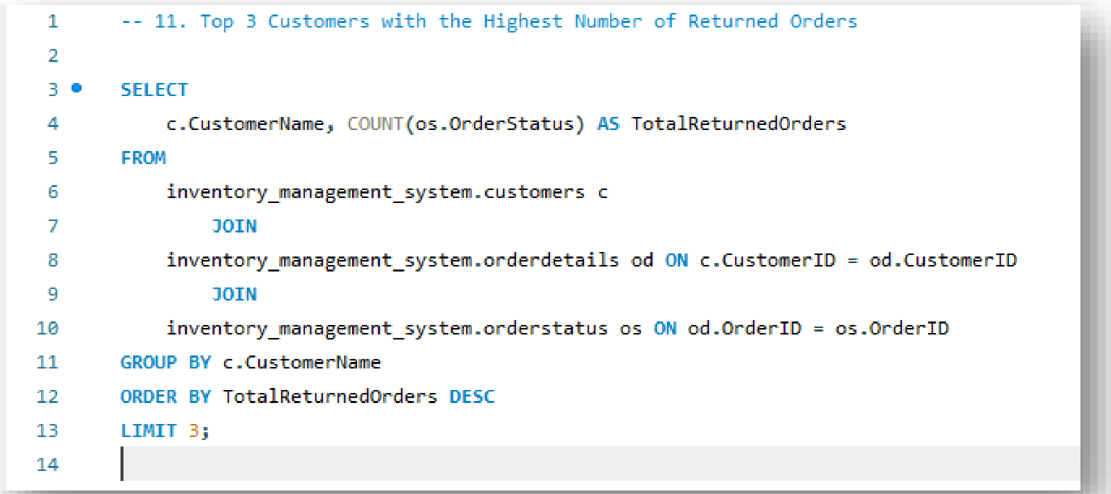


ProductName	TotalQuantitySold	TotalProfit
SmartSole Fitness Shoes	1	67.50

Result 1

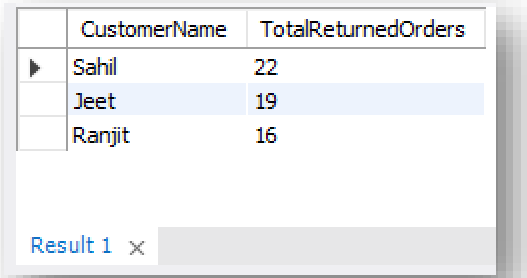
11. Top 3 Customers with the Highest Number of Returned Orders

MySQL Query



```sql
-- 11. Top 3 Customers with the Highest Number of Returned Orders

SELECT
    c.CustomerName, COUNT(os.OrderStatus) AS TotalReturnedOrders
FROM
    inventory_management_system.customers c
        JOIN
    inventory_management_system.orderdetails od ON c.CustomerID = od.CustomerID
        JOIN
    inventory_management_system.orderstatus os ON od.OrderID = os.OrderID
GROUP BY c.CustomerName
ORDER BY TotalReturnedOrders DESC
LIMIT 3;
```

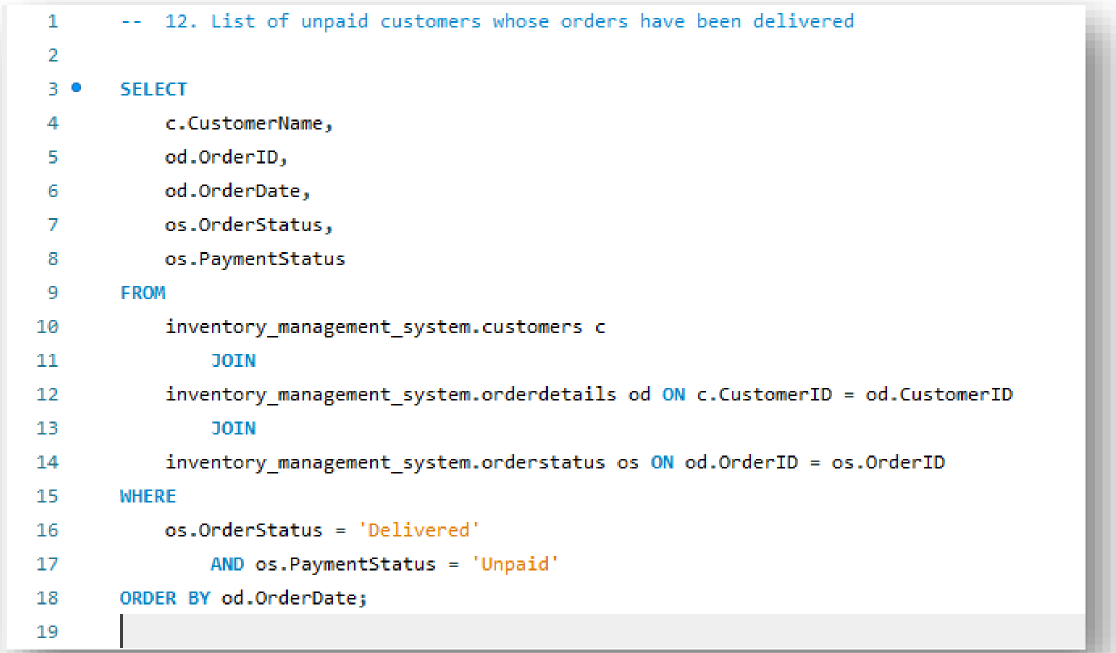
Its Result



CustomerName	TotalReturnedOrders
Sahil	22
Jeet	19
Ranjit	16

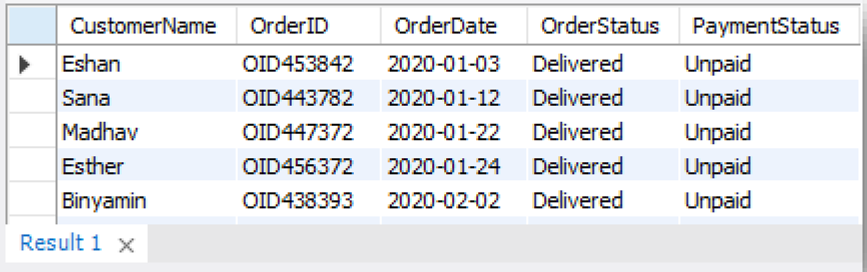
12. List of unpaid customers whose orders have been delivered

MySQL Query



```sql
-- 12. List of unpaid customers whose orders have been delivered

SELECT
    c.CustomerName,
    od.OrderID,
    od.OrderDate,
    os.OrderStatus,
    os.PaymentStatus
FROM
    inventory_management_system.customers c
        JOIN
    inventory_management_system.orderdetails od ON c.CustomerID = od.CustomerID
        JOIN
    inventory_management_system.orderstatus os ON od.OrderID = os.OrderID
WHERE
    os.OrderStatus = 'Delivered'
        AND os.PaymentStatus = 'Unpaid'
ORDER BY od.OrderDate;
```

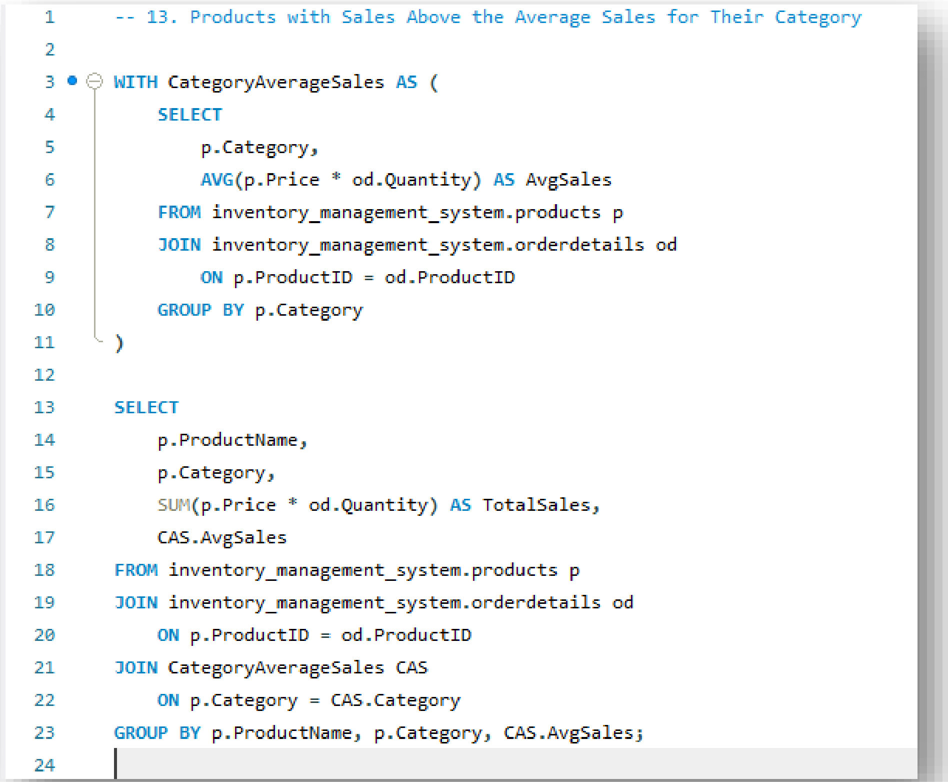
Its Result



CustomerName	OrderID	OrderDate	OrderStatus	PaymentStatus
Eshan	OID453842	2020-01-03	Delivered	Unpaid
Sana	OID443782	2020-01-12	Delivered	Unpaid
Madhav	OID447372	2020-01-22	Delivered	Unpaid
Esther	OID456372	2020-01-24	Delivered	Unpaid
Binyamin	OID438393	2020-02-02	Delivered	Unpaid

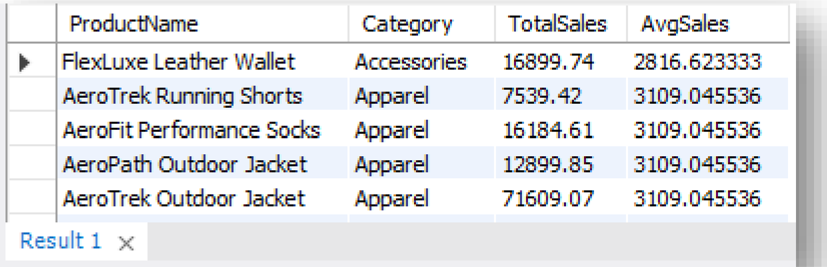
13. Products with Sales Above the Average Sales for Their Category

MySQL Query



```sql
-- 13. Products with Sales Above the Average Sales for Their Category

WITH CategoryAverageSales AS (
    SELECT
        p.Category,
        AVG(p.Price * od.Quantity) AS AvgSales
    FROM inventory_management_system.products p
    JOIN inventory_management_system.orderdetails od
        ON p.ProductID = od.ProductID
    GROUP BY p.Category
)

SELECT
    p.ProductName,
    p.Category,
    SUM(p.Price * od.Quantity) AS TotalSales,
    CAS.AvgSales
FROM inventory_management_system.products p
JOIN inventory_management_system.orderdetails od
    ON p.ProductID = od.ProductID
JOIN CategoryAverageSales CAS
    ON p.Category = CAS.Category
GROUP BY p.ProductName, p.Category, CAS.AvgSales;
```

Its Result

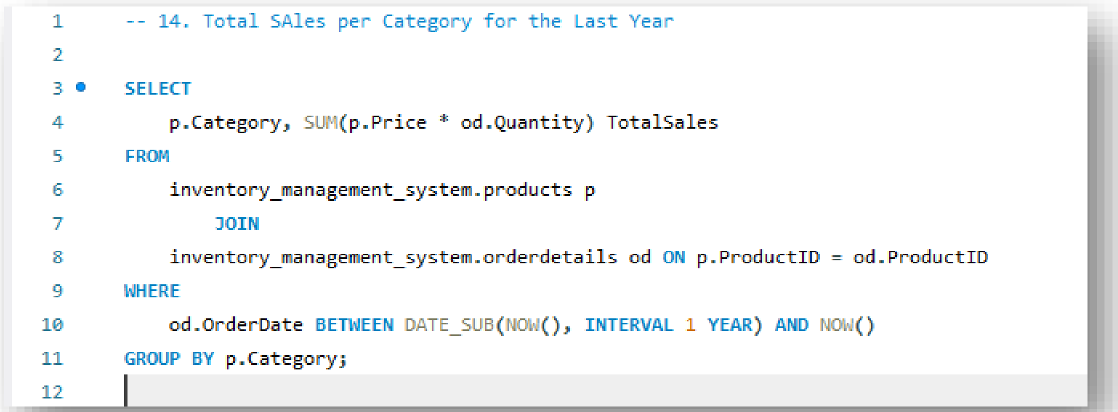


ProductName	Category	TotalSales	AvgSales
FlexLuxe Leather Wallet	Accessories	16899.74	2816.623333
AeroTrek Running Shorts	Apparel	7539.42	3109.045536
AeroFit Performance Socks	Apparel	16184.61	3109.045536
AeroPath Outdoor Jacket	Apparel	12899.85	3109.045536
AeroTrek Outdoor Jacket	Apparel	71609.07	3109.045536

Result 1

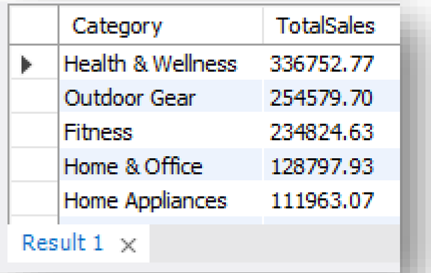
14. Total Sales per Category for the Last Year

MySQL Query



```
-- 14. Total SAles per Category for the Last Year

SELECT
    p.Category, SUM(p.Price * od.Quantity) TotalSales
FROM
    inventory_management_system.products p
        JOIN
    inventory_management_system.orderdetails od ON p.ProductID = od.ProductID
WHERE
    od.OrderDate BETWEEN DATE_SUB(NOW(), INTERVAL 1 YEAR) AND NOW()
GROUP BY p.Category;
```

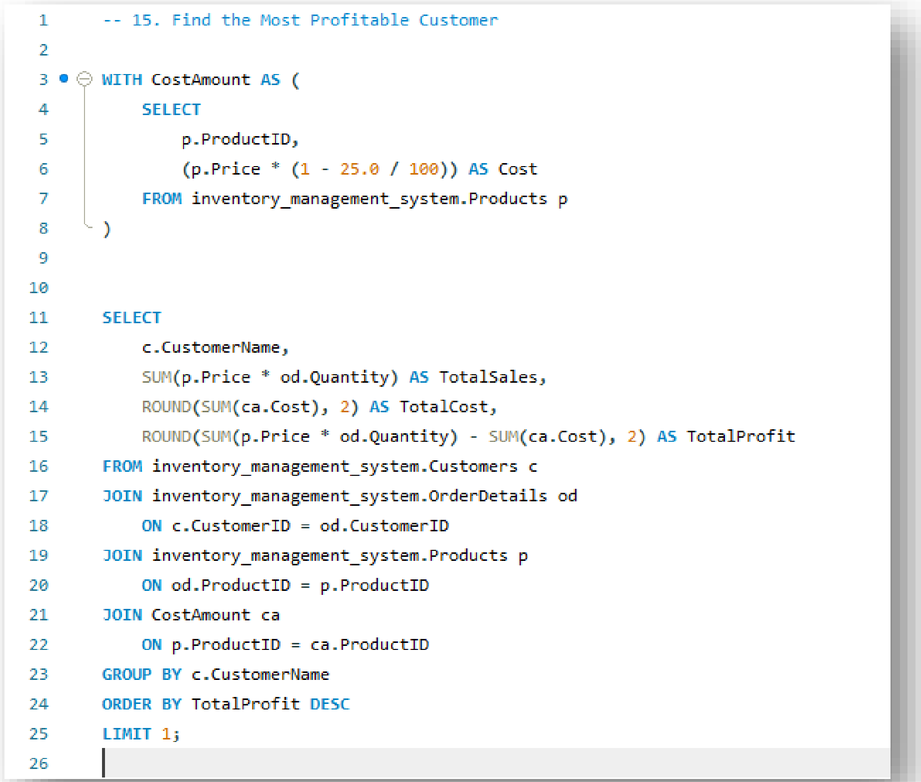
Its Result



Category	TotalSales
Health & Wellness	336752.77
Outdoor Gear	254579.70
Fitness	234824.63
Home & Office	128797.93
Home Appliances	111963.07

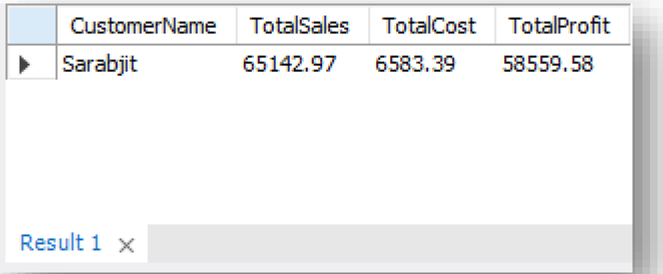
15. Find the Most Profitable Customer

MySQL Query



```
-- 15. Find the Most Profitable Customer

WITH CostAmount AS (
    SELECT
        p.ProductID,
        (p.Price * (1 - 25.0 / 100)) AS Cost
    FROM inventory_management_system.Products p
)


SELECT
    c.CustomerName,
    SUM(p.Price * od.Quantity) AS TotalSales,
    ROUND(SUM(ca.Cost), 2) AS TotalCost,
    ROUND(SUM(p.Price * od.Quantity) - SUM(ca.Cost), 2) AS TotalProfit
FROM inventory_management_system.Customers c
JOIN inventory_management_system.OrderDetails od
    ON c.CustomerID = od.CustomerID
JOIN inventory_management_system.Products p
    ON od.ProductID = p.ProductID
JOIN CostAmount ca
    ON p.ProductID = ca.ProductID
GROUP BY c.CustomerName
ORDER BY TotalProfit DESC
LIMIT 1;
```

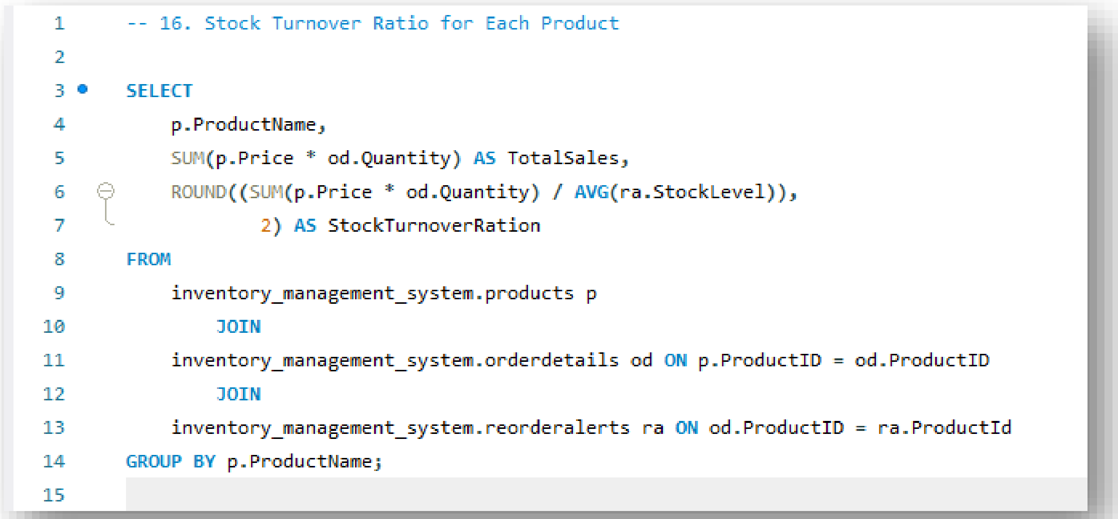
Its Result



CustomerName	TotalSales	TotalCost	TotalProfit
Sarabjit	65142.97	6583.39	58559.58

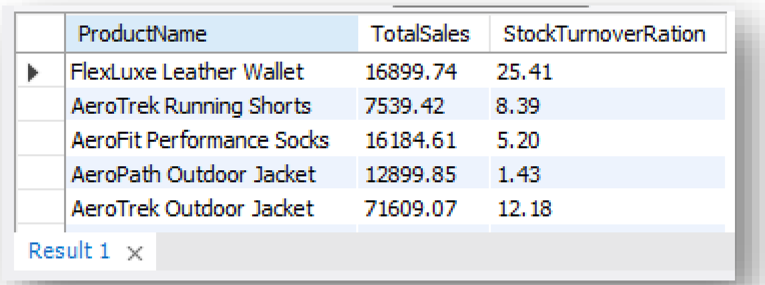
16. Stock Turnover Ratio for Each Product

MySQL Query



```
-- 16. Stock Turnover Ratio for Each Product

SELECT
    p.ProductName,
    SUM(p.Price * od.Quantity) AS TotalSales,
    ROUND((SUM(p.Price * od.Quantity) / AVG(ra.StockLevel)),
            2) AS StockTurnoverRation
FROM
    inventory_management_system.products p
        JOIN
    inventory_management_system.orderdetails od ON p.ProductID = od.ProductID
        JOIN
    inventory_management_system.reorderalerts ra ON od.ProductID = ra.ProductId
GROUP BY p.ProductName;
```

Its Result

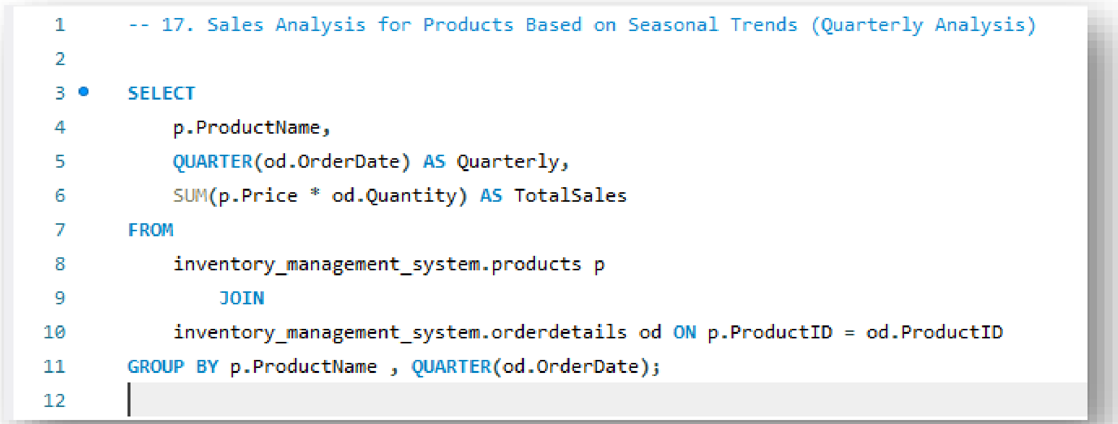


ProductName	TotalSales	StockTurnoverRation
FlexLuxe Leather Wallet	16899.74	25.41
AeroTrek Running Shorts	7539.42	8.39
AeroFit Performance Socks	16184.61	5.20
AeroPath Outdoor Jacket	12899.85	1.43
AeroTrek Outdoor Jacket	71609.07	12.18

Result 1

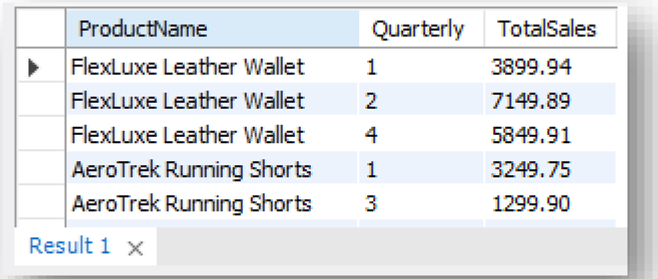
17. Sales Analysis for Products Based on Seasonal Trends (Quarterly Analysis)

MySQL Query



```sql
-- 17. Sales Analysis for Products Based on Seasonal Trends (Quarterly Analysis)

SELECT
    p.ProductName,
    QUARTER(od.OrderDate) AS Quarterly,
    SUM(p.Price * od.Quantity) AS TotalSales
FROM
    inventory_management_system.products p
        JOIN
    inventory_management_system.orderdetails od ON p.ProductID = od.ProductID
GROUP BY p.ProductName , QUARTER(od.OrderDate);
```

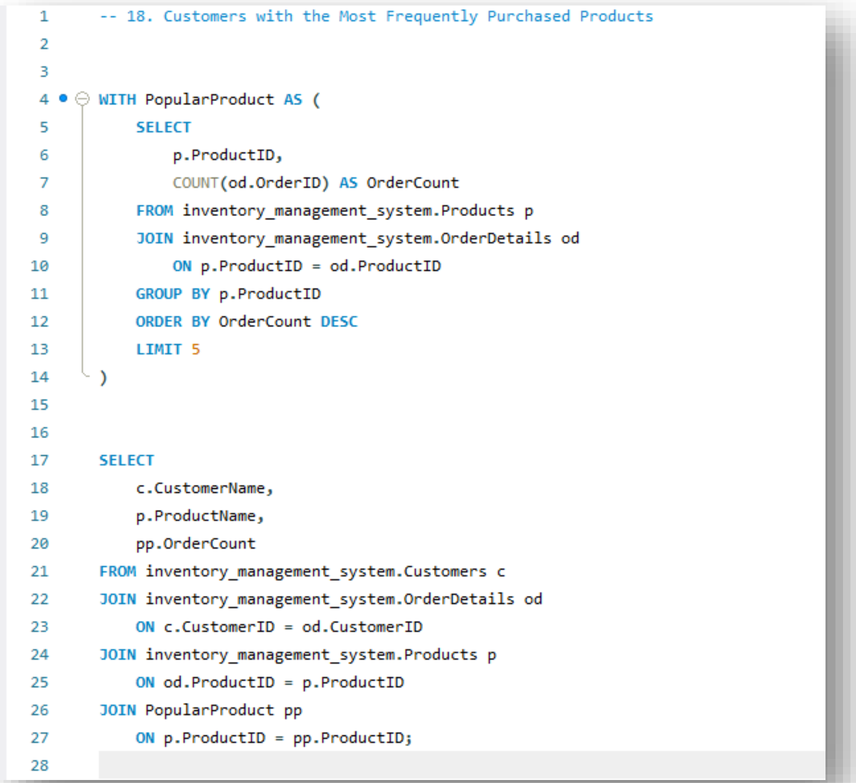
Its Result



ProductName	Quarterly	TotalSales
FlexLuxe Leather Wallet	1	3899.94
FlexLuxe Leather Wallet	2	7149.89
FlexLuxe Leather Wallet	4	5849.91
AeroTrek Running Shorts	1	3249.75
AeroTrek Running Shorts	3	1299.90

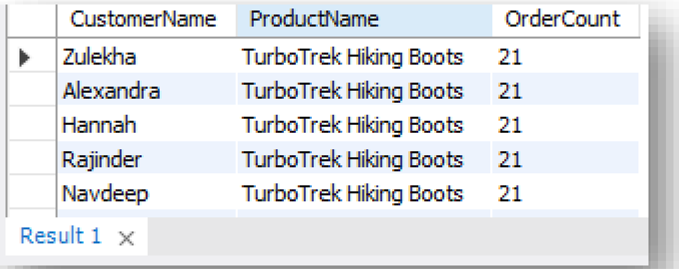
18. Customers with the Most Frequently Purchased Products

MySQL Query



```sql
-- 18. Customers with the Most Frequently Purchased Products


WITH PopularProduct AS (
    SELECT
        p.ProductID,
        COUNT(od.OrderID) AS OrderCount
    FROM inventory_management_system.Products p
    JOIN inventory_management_system.OrderDetails od
        ON p.ProductID = od.ProductID
    GROUP BY p.ProductID
    ORDER BY OrderCount DESC
    LIMIT 5
)


SELECT
    c.CustomerName,
    p.ProductName,
    pp.OrderCount
FROM inventory_management_system.Customers c
JOIN inventory_management_system.OrderDetails od
    ON c.CustomerID = od.CustomerID
JOIN inventory_management_system.Products p
    ON od.ProductID = p.ProductID
JOIN PopularProduct pp
    ON p.ProductID = pp.ProductID;

```

Its Result

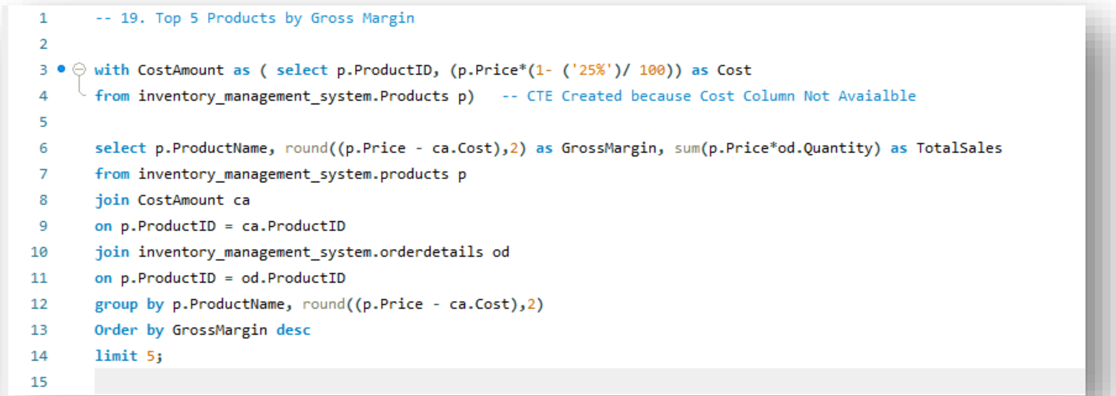


CustomerName	ProductName	OrderCount
Zulekha	TurboTrek Hiking Boots	21
Alexandra	TurboTrek Hiking Boots	21
Hannah	TurboTrek Hiking Boots	21
Rajinder	TurboTrek Hiking Boots	21
Navdeep	TurboTrek Hiking Boots	21

Result 1

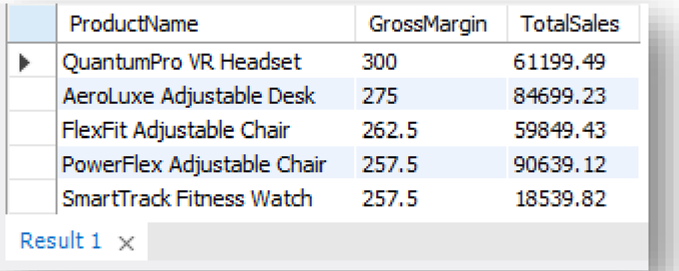
19. Top 5 Products by Gross Margin

MySQL Query



```sql
-- 19. Top 5 Products by Gross Margin

with CostAmount as ( select p.ProductID, (p.Price*(1- ('25%')/ 100)) as Cost
from inventory_management_system.Products p)   -- CTE Created because Cost Column Not Avaialble

select p.ProductName, round((p.Price - ca.Cost),2) as GrossMargin, sum(p.Price*od.Quantity) as TotalSales
from inventory_management_system.products p
join CostAmount ca
on p.ProductID = ca.ProductID
join inventory_management_system.orderdetails od
on p.ProductID = od.ProductID
group by p.ProductName, round((p.Price - ca.Cost),2)
Order by GrossMargin desc
limit 5;

```

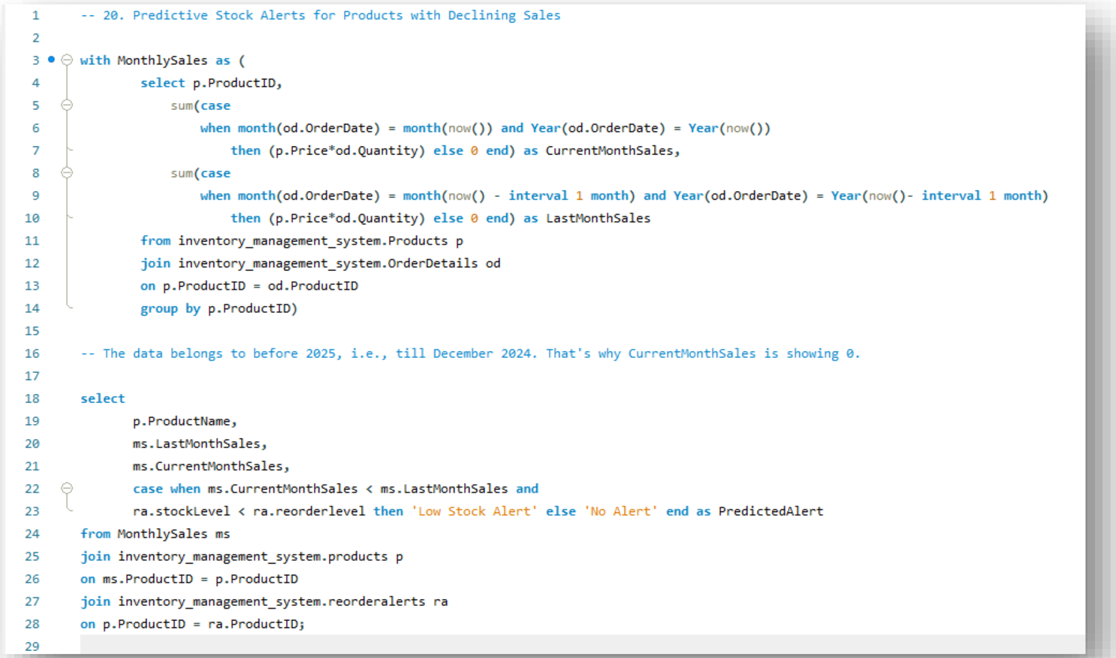
Its Result



ProductName	GrossMargin	TotalSales
QuantumPro VR Headset	300	61199.49
AeroLuxe Adjustable Desk	275	84699.23
FlexFit Adjustable Chair	262.5	59849.43
PowerFlex Adjustable Chair	257.5	90639.12
SmartTrack Fitness Watch	257.5	18539.82

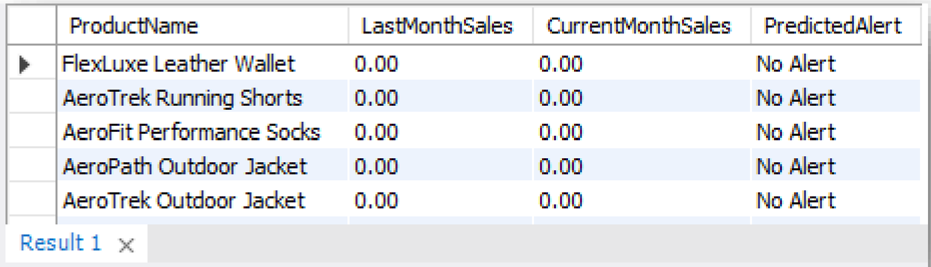
20. Predictive Stock Alerts for Products with Declining Sales

MySQL Query



```sql
-- 20. Predictive Stock Alerts for Products with Declining Sales

with MonthlySales as (
        select p.ProductID,
            sum(case
                when month(od.OrderDate) = month(now()) and Year(od.OrderDate) = Year(now())
                    then (p.Price*od.Quantity) else 0 end) as CurrentMonthSales,
            sum(case
                when month(od.OrderDate) = month(now() - interval 1 month) and Year(od.OrderDate) = Year(now()- interval 1 month)
                    then (p.Price*od.Quantity) else 0 end) as LastMonthSales
        from inventory_management_system.Products p
        join inventory_management_system.OrderDetails od
        on p.ProductID = od.ProductID
        group by p.ProductID)

-- The data belongs to before 2025, i.e., till December 2024. That's why CurrentMonthSales is showing 0.

select
       p.ProductName,
       ms.LastMonthSales,
       ms.CurrentMonthSales,
       case when ms.CurrentMonthSales < ms.LastMonthSales and
       ra.stockLevel < ra.reorderlevel then 'Low Stock Alert' else 'No Alert' end as PredictedAlert
from MonthlySales ms
join inventory_management_system.products p
on ms.ProductID = p.ProductID
join inventory_management_system.reorderalerts ra
on p.ProductID = ra.ProductID;
```

Its Result

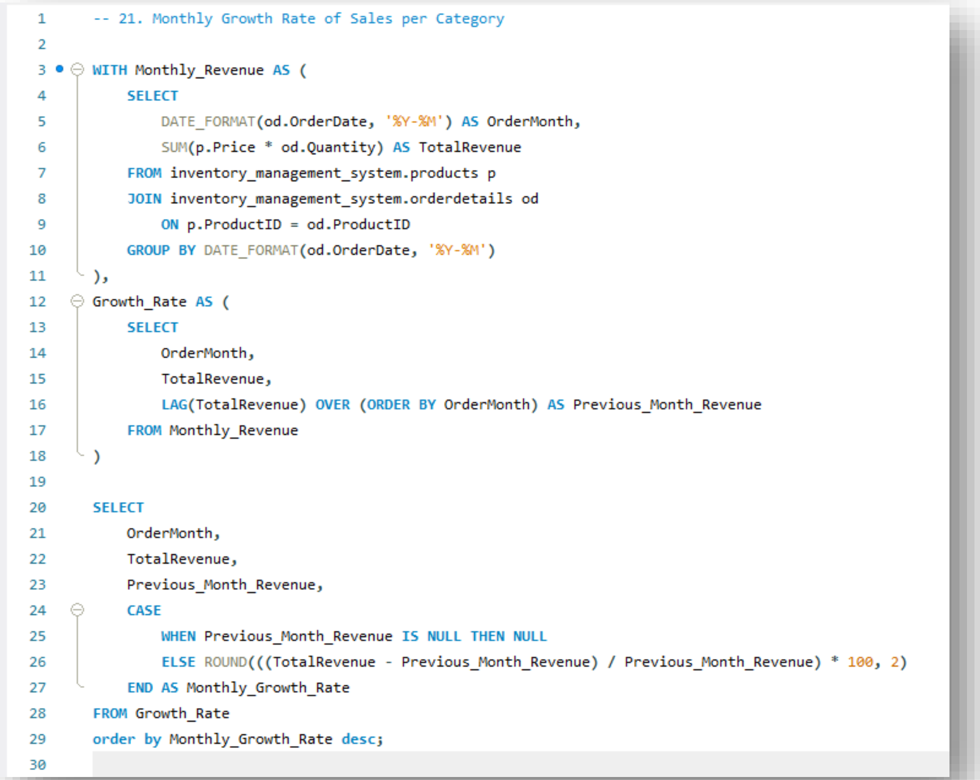


ProductName	LastMonthSales	CurrentMonthSales	PredictedAlert
FlexLuxe Leather Wallet	0.00	0.00	No Alert
AeroTrek Running Shorts	0.00	0.00	No Alert
AeroFit Performance Socks	0.00	0.00	No Alert
AeroPath Outdoor Jacket	0.00	0.00	No Alert
AeroTrek Outdoor Jacket	0.00	0.00	No Alert

Result 1

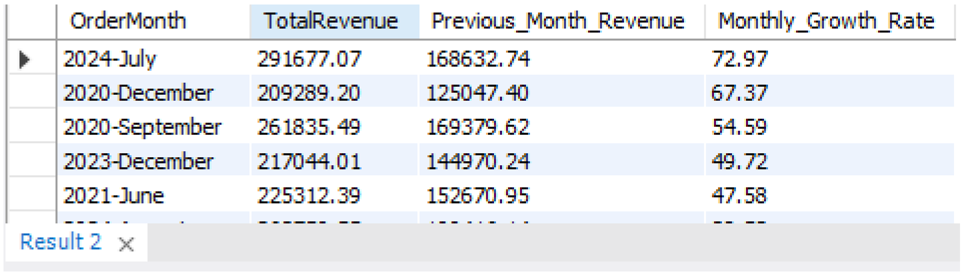
21. Monthly Growth Rate of Sales per Category

MySQL Query



```sql
-- 21. Monthly Growth Rate of Sales per Category

WITH Monthly_Revenue AS (
    SELECT
        DATE_FORMAT(od.OrderDate, '%Y-%M') AS OrderMonth,
        SUM(p.Price * od.Quantity) AS TotalRevenue
    FROM inventory_management_system.products p
    JOIN inventory_management_system.orderdetails od
        ON p.ProductID = od.ProductID
    GROUP BY DATE_FORMAT(od.OrderDate, '%Y-%M')
),
Growth_Rate AS (
    SELECT
        OrderMonth,
        TotalRevenue,
        LAG(TotalRevenue) OVER (ORDER BY OrderMonth) AS Previous_Month_Revenue
    FROM Monthly_Revenue
)

SELECT
    OrderMonth,
    TotalRevenue,
    Previous_Month_Revenue,
    CASE
        WHEN Previous_Month_Revenue IS NULL THEN NULL
        ELSE ROUND(((TotalRevenue - Previous_Month_Revenue) / Previous_Month_Revenue) * 100, 2)
    END AS Monthly_Growth_Rate
FROM Growth_Rate
order by Monthly_Growth_Rate desc;
```

Its Result

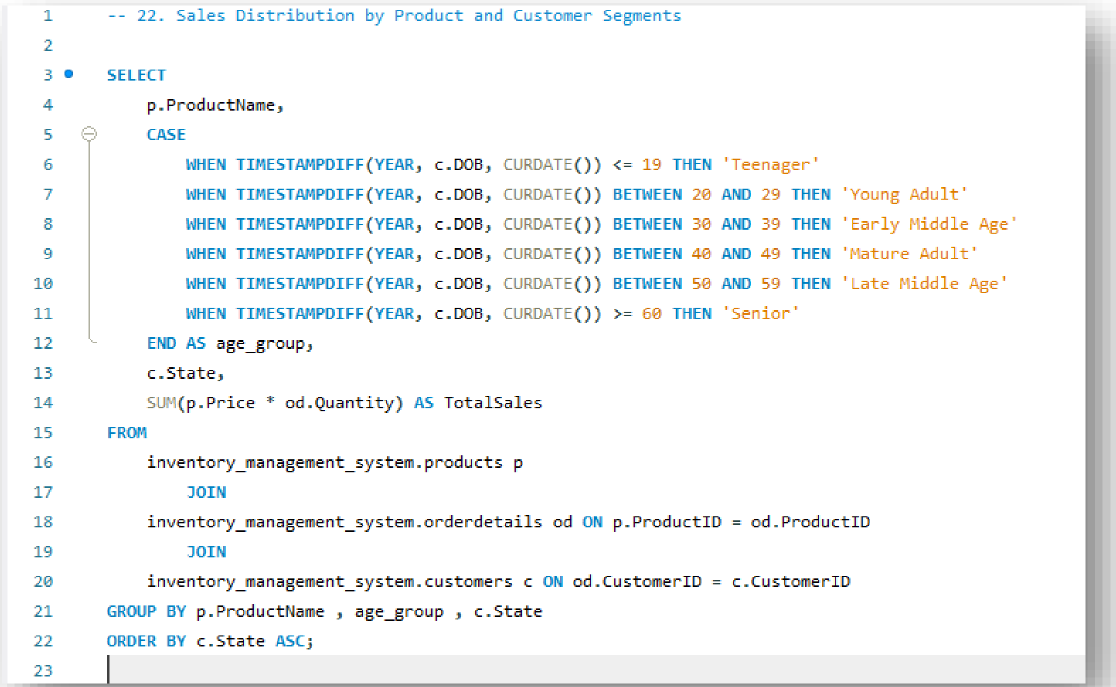


OrderMonth	TotalRevenue	Previous_Month_Revenue	Monthly_Growth_Rate
2024-July	291677.07	168632.74	72.97
2020-December	209289.20	125047.40	67.37
2020-September	261835.49	169379.62	54.59
2023-December	217044.01	144970.24	49.72
2021-June	225312.39	152670.95	47.58

Result 2

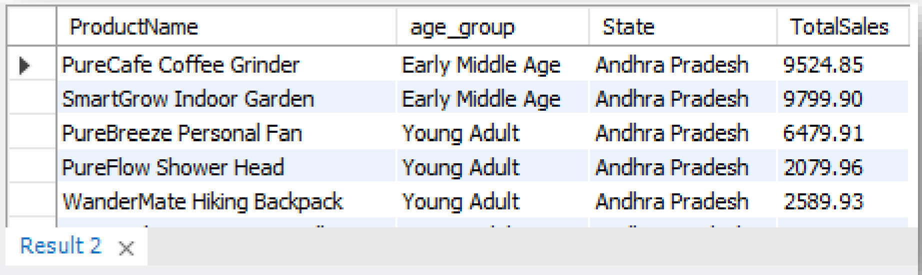
22. Sales Distribution by Product and Customer Segments

MySQL Query



```sql
-- 22. Sales Distribution by Product and Customer Segments

SELECT
    p.ProductName,
    CASE
        WHEN TIMESTAMPDIFF(YEAR, c.DOB, CURDATE()) <= 19 THEN 'Teenager'
        WHEN TIMESTAMPDIFF(YEAR, c.DOB, CURDATE()) BETWEEN 20 AND 29 THEN 'Young Adult'
        WHEN TIMESTAMPDIFF(YEAR, c.DOB, CURDATE()) BETWEEN 30 AND 39 THEN 'Early Middle Age'
        WHEN TIMESTAMPDIFF(YEAR, c.DOB, CURDATE()) BETWEEN 40 AND 49 THEN 'Mature Adult'
        WHEN TIMESTAMPDIFF(YEAR, c.DOB, CURDATE()) BETWEEN 50 AND 59 THEN 'Late Middle Age'
        WHEN TIMESTAMPDIFF(YEAR, c.DOB, CURDATE()) >= 60 THEN 'Senior'
    END AS age_group,
    c.State,
    SUM(p.Price * od.Quantity) AS TotalSales
FROM
    inventory_management_system.products p
        JOIN
    inventory_management_system.orderdetails od ON p.ProductID = od.ProductID
        JOIN
    inventory_management_system.customers c ON od.CustomerID = c.CustomerID
GROUP BY p.ProductName , age_group , c.State
ORDER BY c.State ASC;
```

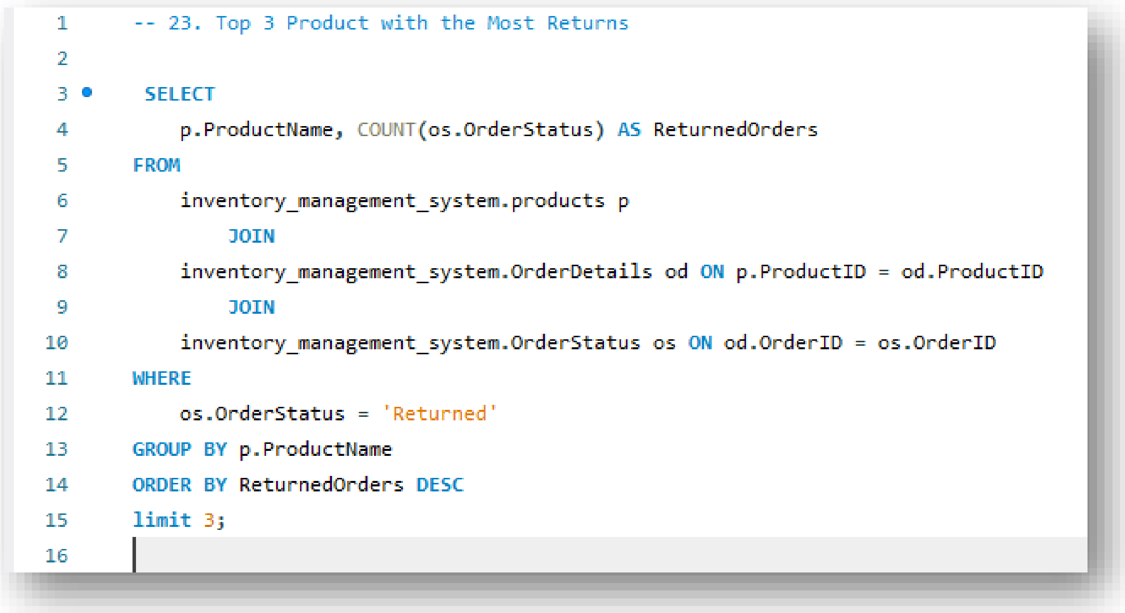
Its Result



ProductName	age_group	State	TotalSales
PureCafe Coffee Grinder	Early Middle Age	Andhra Pradesh	9524.85
SmartGrow Indoor Garden	Early Middle Age	Andhra Pradesh	9799.90
PureBreeze Personal Fan	Young Adult	Andhra Pradesh	6479.91
PureFlow Shower Head	Young Adult	Andhra Pradesh	2079.96
WanderMate Hiking Backpack	Young Adult	Andhra Pradesh	2589.93

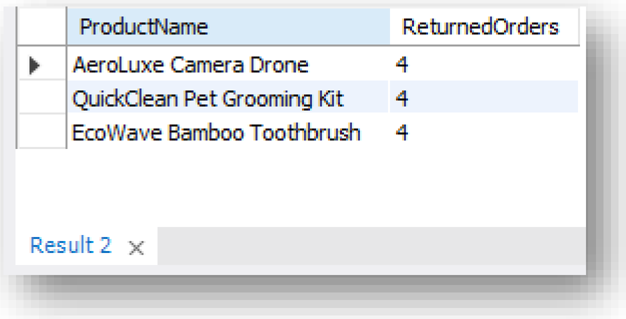
23. Top 3 Product with the Most Returns

MySQL Query



```sql
-- 23. Top 3 Product with the Most Returns

 SELECT
    p.ProductName, COUNT(os.OrderStatus) AS ReturnedOrders
FROM
    inventory_management_system.products p
        JOIN
    inventory_management_system.OrderDetails od ON p.ProductID = od.ProductID
        JOIN
    inventory_management_system.OrderStatus os ON od.OrderID = os.OrderID
WHERE
    os.OrderStatus = 'Returned'
GROUP BY p.ProductName
ORDER BY ReturnedOrders DESC
limit 3;
```

Its Result

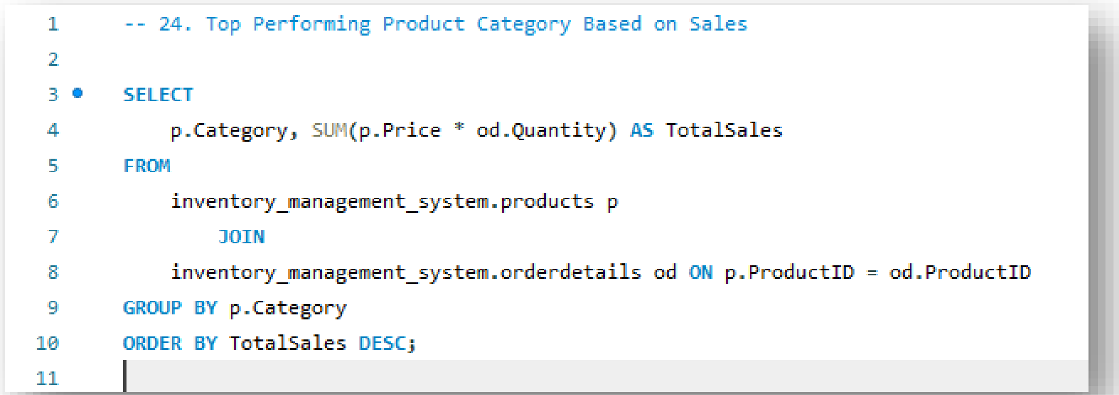


ProductName	ReturnedOrders
AeroLuxe Camera Drone	4
QuickClean Pet Grooming Kit	4
EcoWave Bamboo Toothbrush	4

Result 2

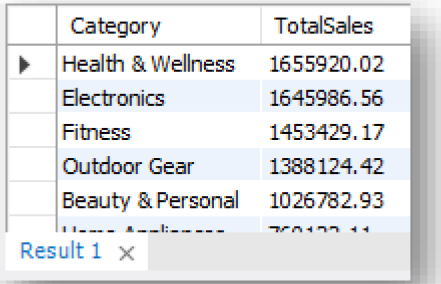
24. Top Performing Product Category Based on Sales

MySQL Query



```sql
-- 24. Top Performing Product Category Based on Sales

SELECT
    p.Category, SUM(p.Price * od.Quantity) AS TotalSales
FROM
    inventory_management_system.products p
        JOIN
    inventory_management_system.orderdetails od ON p.ProductID = od.ProductID
GROUP BY p.Category
ORDER BY TotalSales DESC;

```

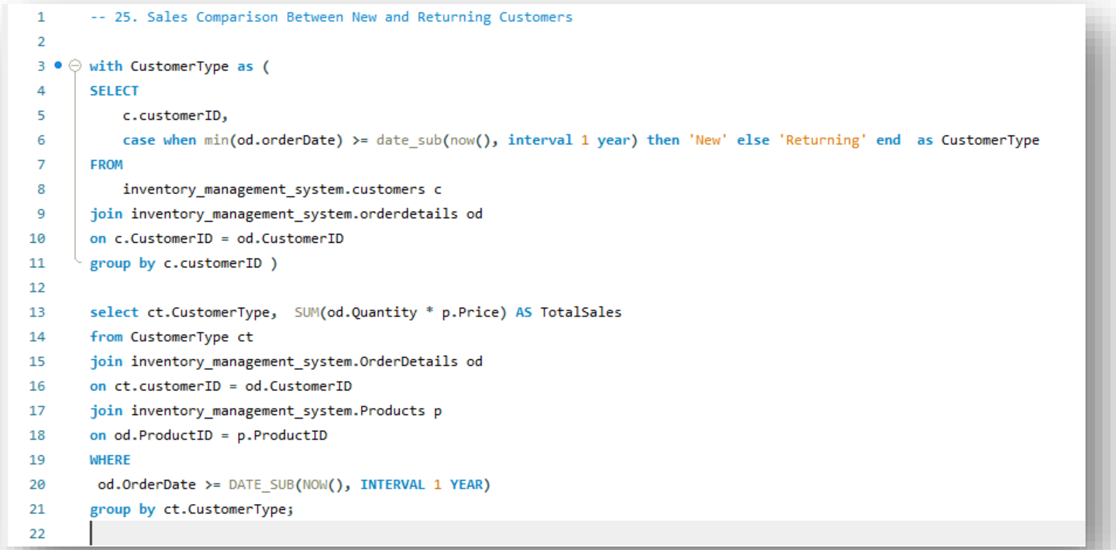
Its Result



Category	TotalSales
Health & Wellness	1655920.02
Electronics	1645986.56
Fitness	1453429.17
Outdoor Gear	1388124.42
Beauty & Personal	1026782.93
[illegible]	[illegible]

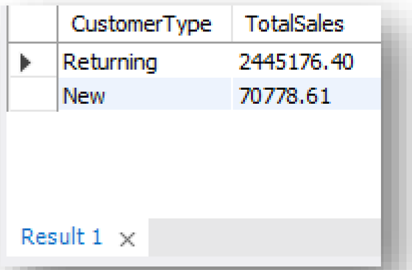
25. Sales Comparison Between New and Returning Customers

MySQL Query



```sql
-- 25. Sales Comparison Between New and Returning Customers

with CustomerType as (
SELECT
    c.customerID,
    case when min(od.orderDate) >= date_sub(now(), interval 1 year) then 'New' else 'Returning' end  as CustomerType
FROM
    inventory_management_system.customers c
join inventory_management_system.orderdetails od
on c.CustomerID = od.CustomerID
group by c.customerID )

select ct.CustomerType,  SUM(od.Quantity * p.Price) AS TotalSales
from CustomerType ct
join inventory_management_system.OrderDetails od
on ct.customerID = od.CustomerID
join inventory_management_system.Products p
on od.ProductID = p.ProductID
WHERE
 od.OrderDate >= DATE_SUB(NOW(), INTERVAL 1 YEAR)
group by ct.CustomerType;
```

Its Result

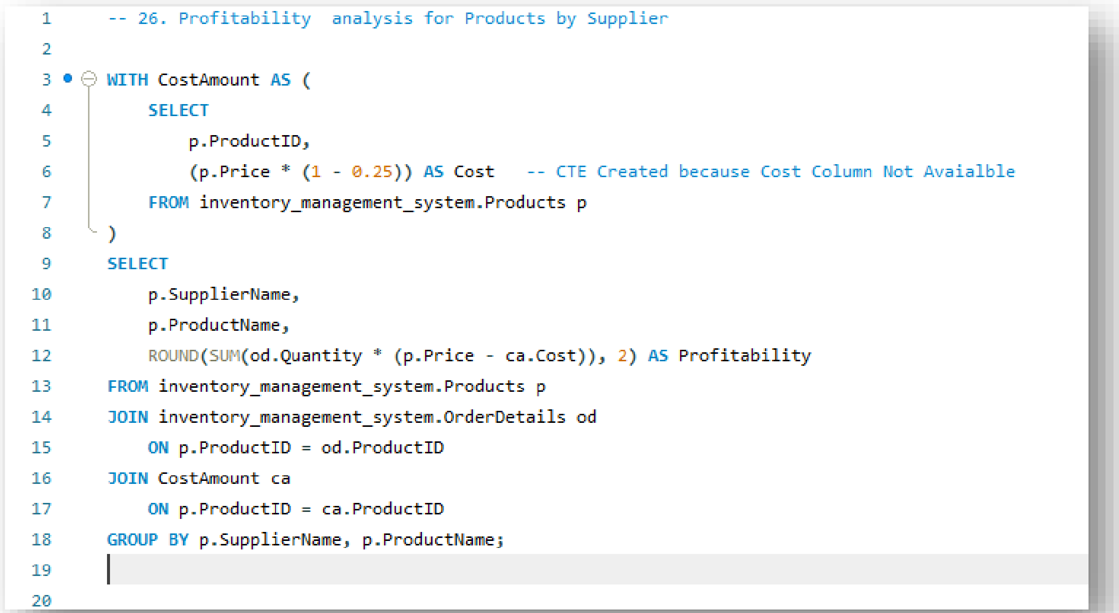


CustomerType	TotalSales
Returning	2445176.40
New	70778.61

Result 1

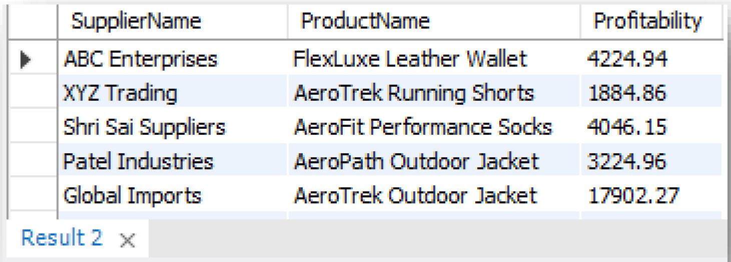
26. Profitability analysis for Products by Supplier

MySQL Query



```sql
-- 26. Profitability  analysis for Products by Supplier

WITH CostAmount AS (
    SELECT
        p.ProductID,
        (p.Price * (1 - 0.25)) AS Cost   -- CTE Created because Cost Column Not Avaialble
    FROM inventory_management_system.Products p
)
SELECT
    p.SupplierName,
    p.ProductName,
    ROUND(SUM(od.Quantity * (p.Price - ca.Cost)), 2) AS Profitability
FROM inventory_management_system.Products p
JOIN inventory_management_system.OrderDetails od
    ON p.ProductID = od.ProductID
JOIN CostAmount ca
    ON p.ProductID = ca.ProductID
GROUP BY p.SupplierName, p.ProductName;
```

Its Result



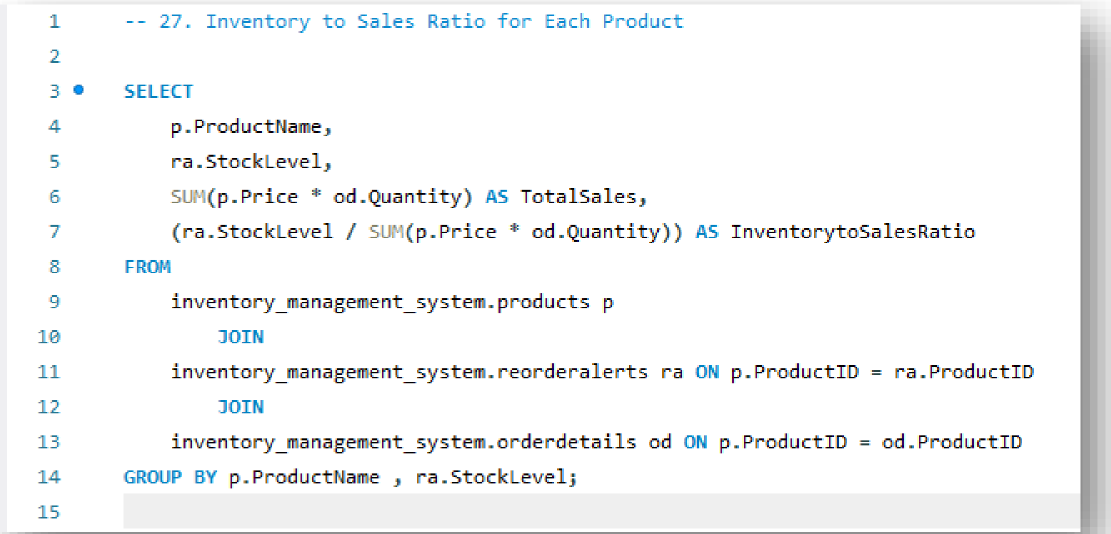
SupplierName	ProductName	Profitability
ABC Enterprises	FlexLuxe Leather Wallet	4224.94
XYZ Trading	AeroTrek Running Shorts	1884.86
Shri Sai Suppliers	AeroFit Performance Socks	4046.15
Patel Industries	AeroPath Outdoor Jacket	3224.96
Global Imports	AeroTrek Outdoor Jacket	17902.27

Result 2

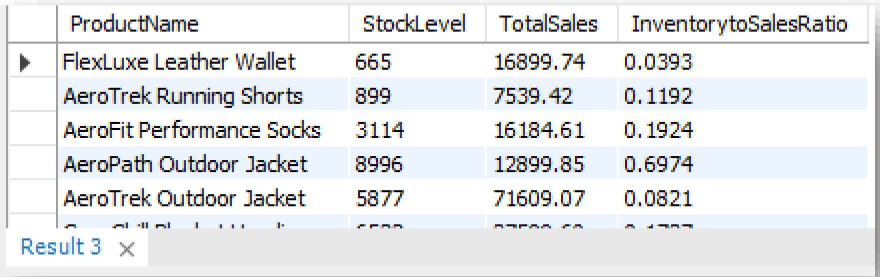
27. Inventory to Sales Ratio for Each Product

MySQL Query

Its Result



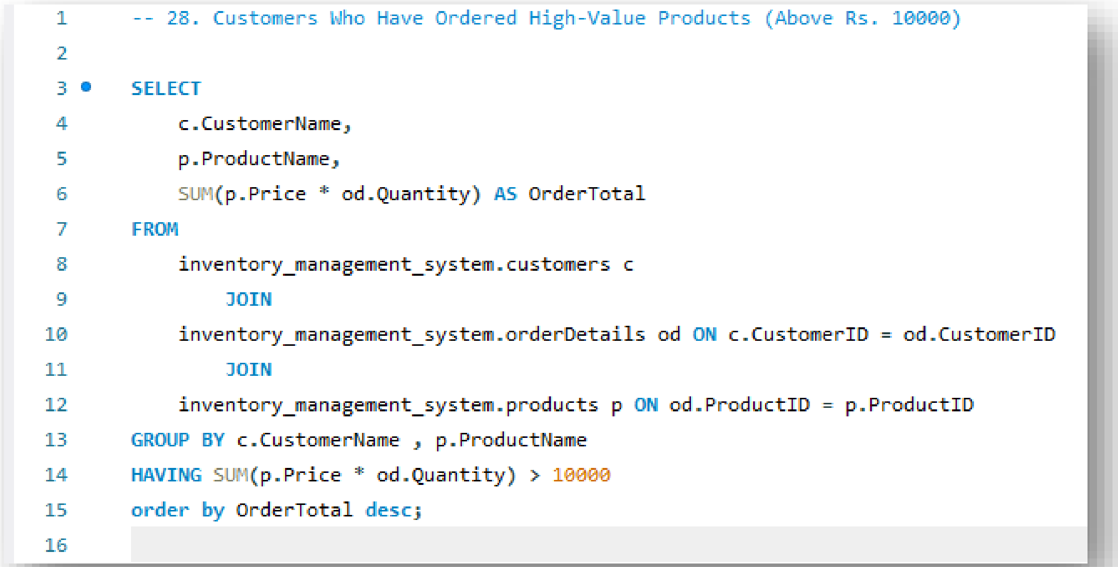
```sql
-- 27. Inventory to Sales Ratio for Each Product

SELECT
    p.ProductName,
    ra.StockLevel,
    SUM(p.Price * od.Quantity) AS TotalSales,
    (ra.StockLevel / SUM(p.Price * od.Quantity)) AS InventorytoSalesRatio
FROM
    inventory_management_system.products p
        JOIN
    inventory_management_system.reorderalerts ra ON p.ProductID = ra.ProductID
        JOIN
    inventory_management_system.orderdetails od ON p.ProductID = od.ProductID
GROUP BY p.ProductName , ra.StockLevel;
```



ProductName	StockLevel	TotalSales	InventorytoSalesRatio
FlexLuxe Leather Wallet	665	16899.74	0.0393
AeroTrek Running Shorts	899	7539.42	0.1192
AeroFit Performance Socks	3114	16184.61	0.1924
AeroPath Outdoor Jacket	8996	12899.85	0.6974
AeroTrek Outdoor Jacket	5877	71609.07	0.0821
[illegible]	[illegible]	[illegible]	[illegible]

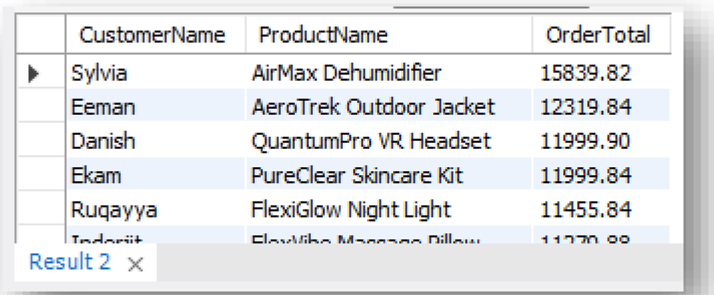
28. Customers Who Have Ordered High-Value Products (Above Rs. 10000)

MySQL Query



```sql
-- 28. Customers Who Have Ordered High-Value Products (Above Rs. 10000)

SELECT
    c.CustomerName,
    p.ProductName,
    SUM(p.Price * od.Quantity) AS OrderTotal
FROM
    inventory_management_system.customers c
        JOIN
    inventory_management_system.orderDetails od ON c.CustomerID = od.CustomerID
        JOIN
    inventory_management_system.products p ON od.ProductID = p.ProductID
GROUP BY c.CustomerName , p.ProductName
HAVING SUM(p.Price * od.Quantity) > 10000
order by OrderTotal desc;

```

Its Result

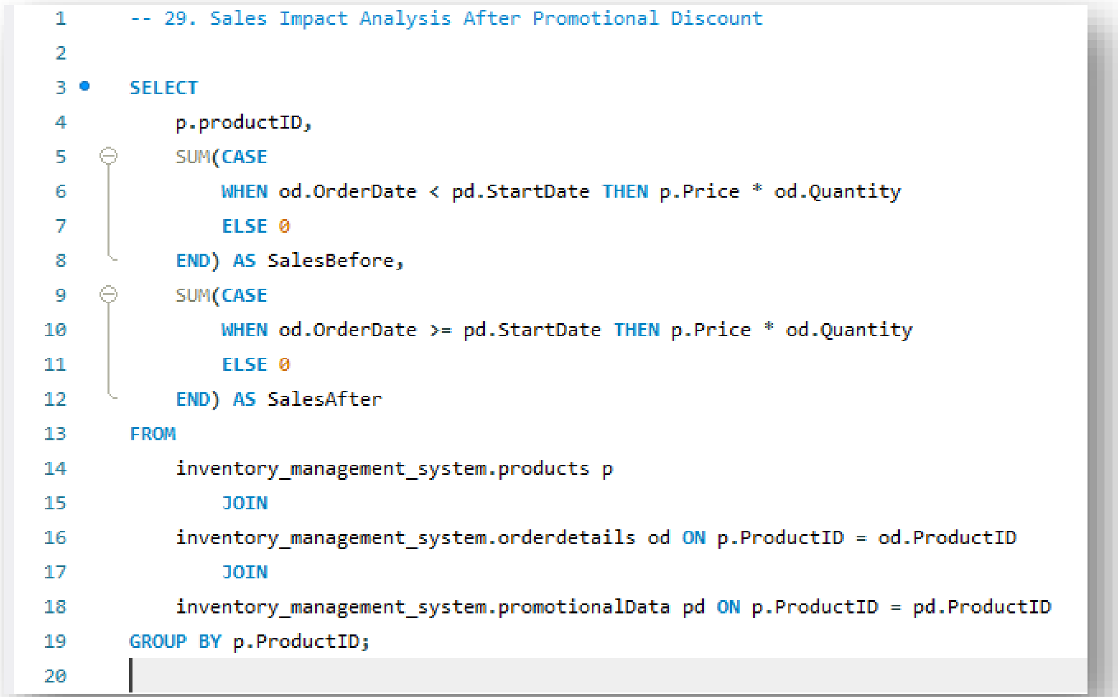


CustomerName	ProductName	OrderTotal
Sylvia	AirMax Dehumidifier	15839.82
Eeman	AeroTrek Outdoor Jacket	12319.84
Danish	QuantumPro VR Headset	11999.90
Ekam	PureClear Skincare Kit	11999.84
Ruqayya	FlexiGlow Night Light	11455.84
Inderjit	FlexVibe Massage Pillow	11279.88

Result 2

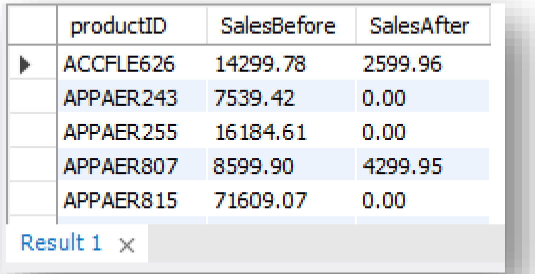
29. Sales Impact Analysis After Promotional Discount

MySQL Query



```
-- 29. Sales Impact Analysis After Promotional Discount

SELECT
    p.productID,
    SUM(CASE
        WHEN od.OrderDate < pd.StartDate THEN p.Price * od.Quantity
        ELSE 0
    END) AS SalesBefore,
    SUM(CASE
        WHEN od.OrderDate >= pd.StartDate THEN p.Price * od.Quantity
        ELSE 0
    END) AS SalesAfter
FROM
    inventory_management_system.products p
        JOIN
    inventory_management_system.orderdetails od ON p.ProductID = od.ProductID
        JOIN
    inventory_management_system.promotionalData pd ON p.ProductID = pd.ProductID
GROUP BY p.ProductID;
```

Its Result

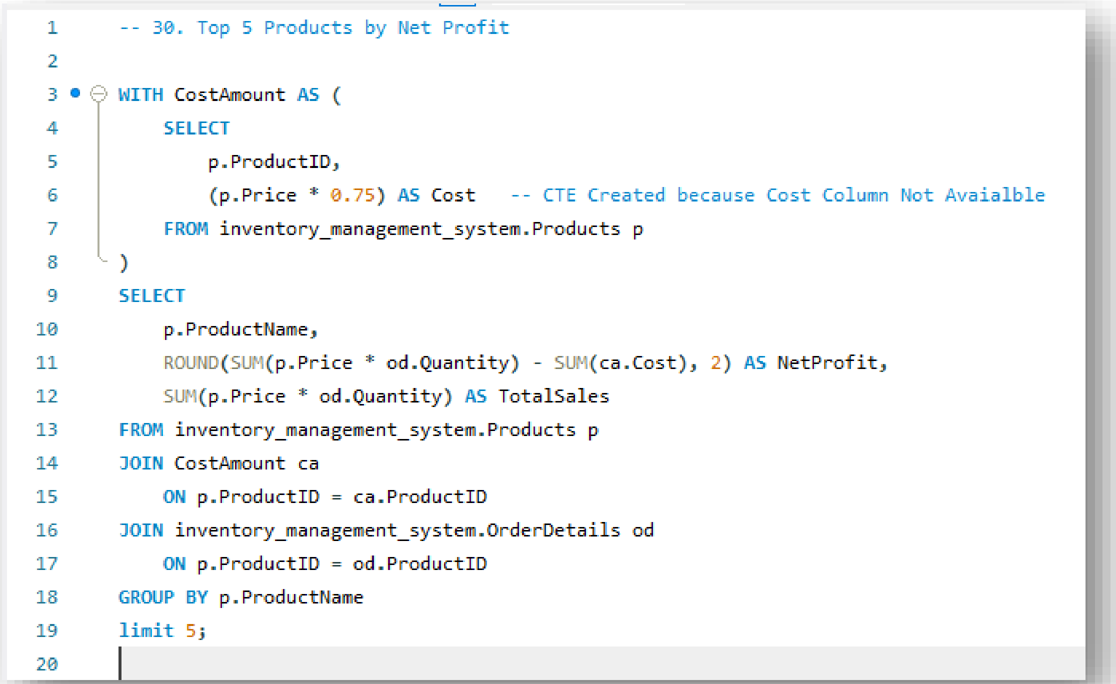


productID	SalesBefore	SalesAfter
ACCFLE626	14299.78	2599.96
APPAER243	7539.42	0.00
APPAER255	16184.61	0.00
APPAER807	8599.90	4299.95
APPAER815	71609.07	0.00

Result 1

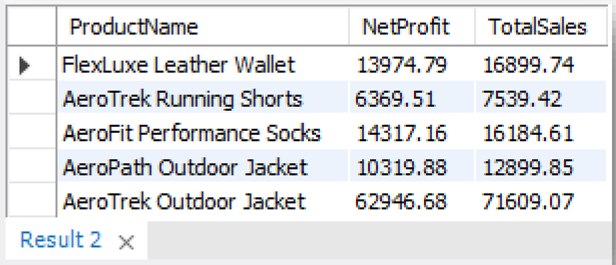
30. Top 5 Products by Net Profit

MySQL Query



```sql
-- 30. Top 5 Products by Net Profit

WITH CostAmount AS (
    SELECT
        p.ProductID,
        (p.Price * 0.75) AS Cost    -- CTE Created because Cost Column Not Avaialble
    FROM inventory_management_system.Products p
)
SELECT
    p.ProductName,
    ROUND(SUM(p.Price * od.Quantity) - SUM(ca.Cost), 2) AS NetProfit,
    SUM(p.Price * od.Quantity) AS TotalSales
FROM inventory_management_system.Products p
JOIN CostAmount ca
    ON p.ProductID = ca.ProductID
JOIN inventory_management_system.OrderDetails od
    ON p.ProductID = od.ProductID
GROUP BY p.ProductName
limit 5;
```

Its Result



ProductName	NetProfit	TotalSales
FlexLuxe Leather Wallet	13974.79	16899.74
AeroTrek Running Shorts	6369.51	7539.42
AeroFit Performance Socks	14317.16	16184.61
AeroPath Outdoor Jacket	10319.88	12899.85
AeroTrek Outdoor Jacket	62946.68	71609.07

Result 2