RETAIL INTELLIGENCE : ADVANCED SQL CASE STUDY ON GLOBAL SUPERSTORE

Anuska – Aspiring Data Analyst
June 2025

EXECUTIVE SUMMARY

OVERVIEW

- OBJECTIVE : Analyze Sales , delivery performance , customer behavior, and profitability patterns using SQL.
- Dataset : **GLOBAL SUPERSTORE** (Orders , Returns , People)
- Business Outcomes :

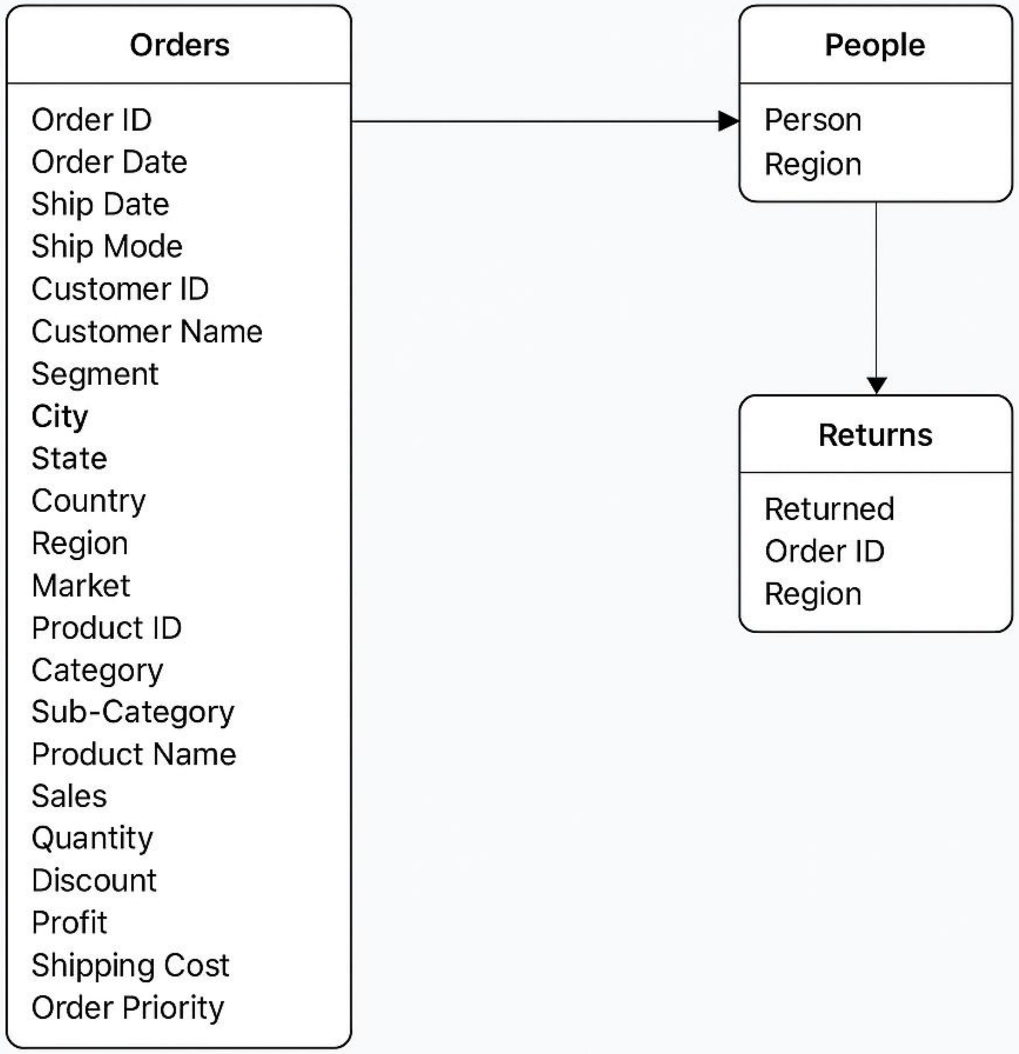
  Identified high-profit customers

  Detected risky product categories

  Segmented customers using RFM

  Analyzed delivery delays and growth trends

: Dataset Overview



Orders

Order ID
Order Date
Ship Date
Ship Mode
Customer ID
Customer Name
Segment
City
State
Country
Region
Market
Product ID
Category
Sub-Category
Product Name
Sales
Quantity
Discount
Profit
Shipping Cost
Order Priority

People

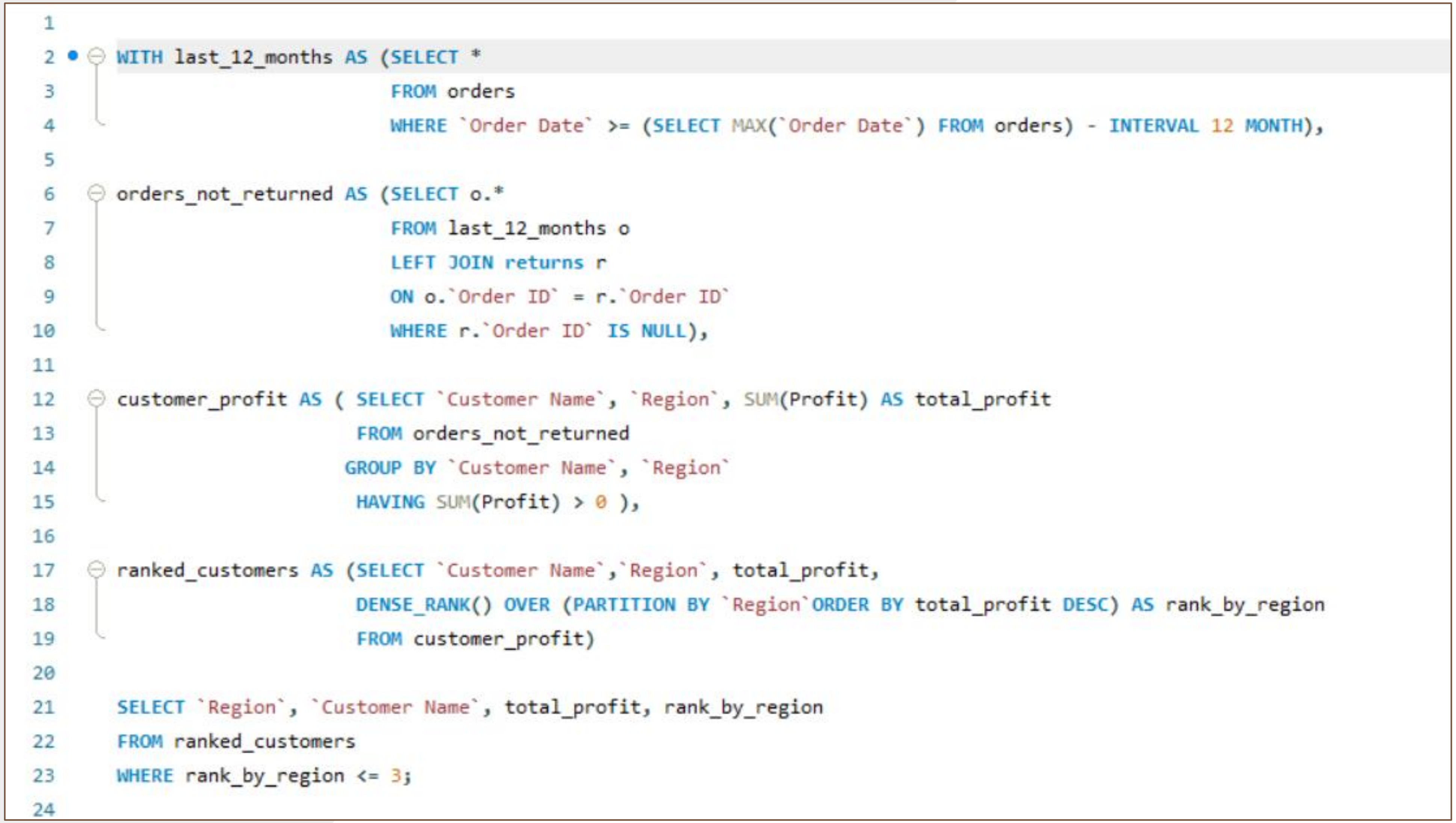
Person
Region

Returns

Returned
Order ID
Region

Schema diagram

QUES 1-- TOP 3 MOST PROFITABLE CUSTOMERS BY REGION (OVER LAST 12 MONTHS)



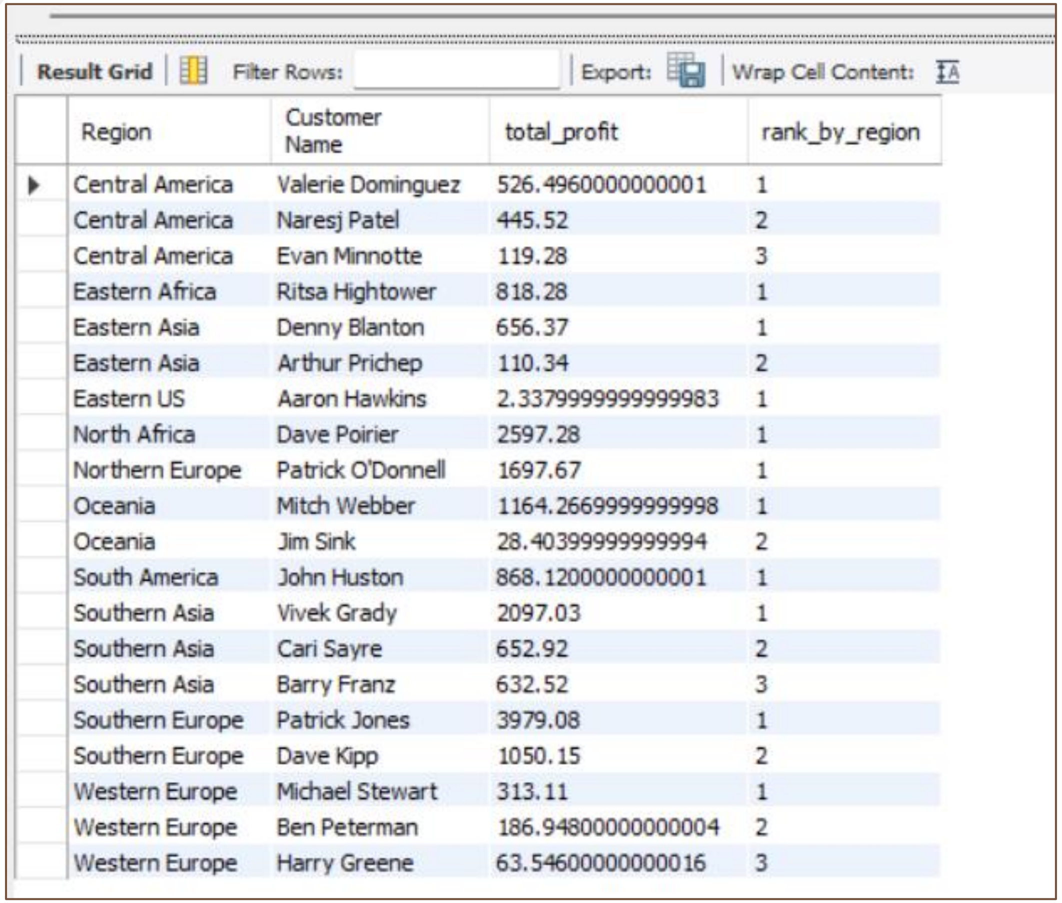
```sql

WITH last_12_months AS (SELECT *
                        FROM orders
                        WHERE `Order Date` >= (SELECT MAX(`Order Date`) FROM orders) - INTERVAL 12 MONTH),

orders_not_returned AS (SELECT o.*
                        FROM last_12_months o
                        LEFT JOIN returns r
                        ON o.`Order ID` = r.`Order ID`
                        WHERE r.`Order ID` IS NULL),

customer_profit AS ( SELECT `Customer Name`, `Region`, SUM(Profit) AS total_profit
                     FROM orders_not_returned
                    GROUP BY `Customer Name`, `Region`
                     HAVING SUM(Profit) > 0 ),

ranked_customers AS (SELECT `Customer Name`,`Region`, total_profit,
                     DENSE_RANK() OVER (PARTITION BY `Region`ORDER BY total_profit DESC) AS rank_by_region
                     FROM customer_profit)

SELECT `Region`, `Customer Name`, total_profit, rank_by_region
FROM ranked_customers
WHERE rank_by_region <= 3;

```

OBJECTIVE: IDENTIFY TOP 3 CUSTOMERS PER REGION BASED ON TOTAL PROFIT (LAST 12 MONTHS).

TECHNIQUES: CTE ,AGGREGATION ,DENSE_RANK().

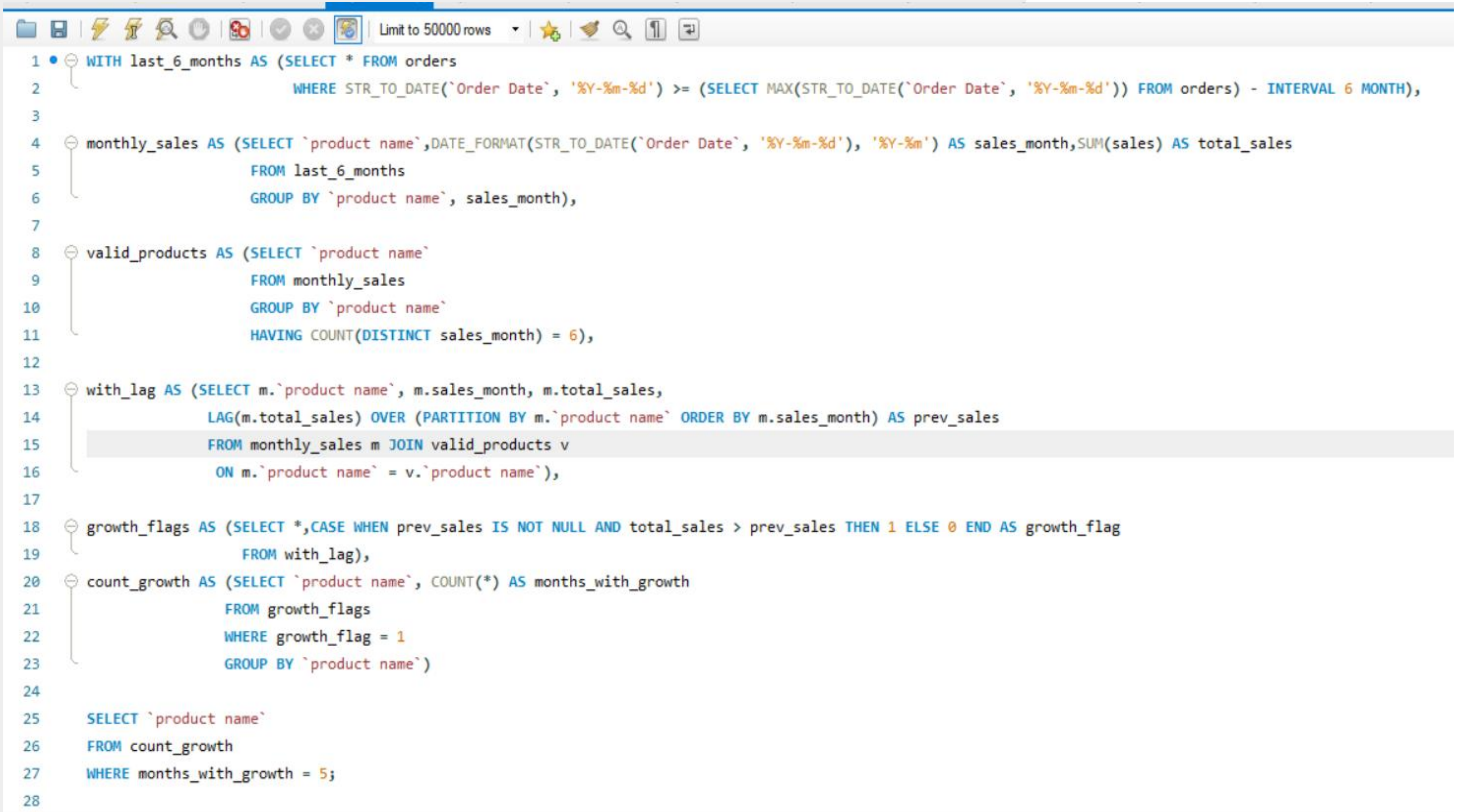
INSIGHT: PROFIT IS HEAVILY DRIVEN BY REPEAT BUYERS IN EACH REGION.

Only customers who placed orders in the **last 12 months** were included. Many regions had more than 3 active customers, but only the **top 3 by profit** were selected per region. Regions with fewer active customers had fewer rows.



Region	Customer Name	total_profit	rank_by_region
Central America	Valerie Dominguez	526.4960000000001	1
Central America	Naresj Patel	445.52	2
Central America	Evan Minnotte	119.28	3
Eastern Africa	Ritsa Hightower	818.28	1
Eastern Asia	Denny Blanton	656.37	1
Eastern Asia	Arthur Prichep	110.34	2
Eastern US	Aaron Hawkins	2.3379999999999983	1
North Africa	Dave Poirier	2597.28	1
Northern Europe	Patrick O'Donnell	1697.67	1
Oceania	Mitch Webber	1164.2669999999998	1
Oceania	Jim Sink	28.403999999999994	2
South America	John Huston	868.1200000000001	1
Southern Asia	Vivek Grady	2097.03	1
Southern Asia	Cari Sayre	652.92	2
Southern Asia	Barry Franz	632.52	3
Southern Europe	Patrick Jones	3979.08	1
Southern Europe	Dave Kipp	1050.15	2
Western Europe	Michael Stewart	313.11	1
Western Europe	Ben Peterman	186.94800000000004	2
Western Europe	Harry Greene	63.54600000000016	3

QUES 2 – PRODUCTS WITH CONTINUOUS SALES GROWTH(6 MONTHS)



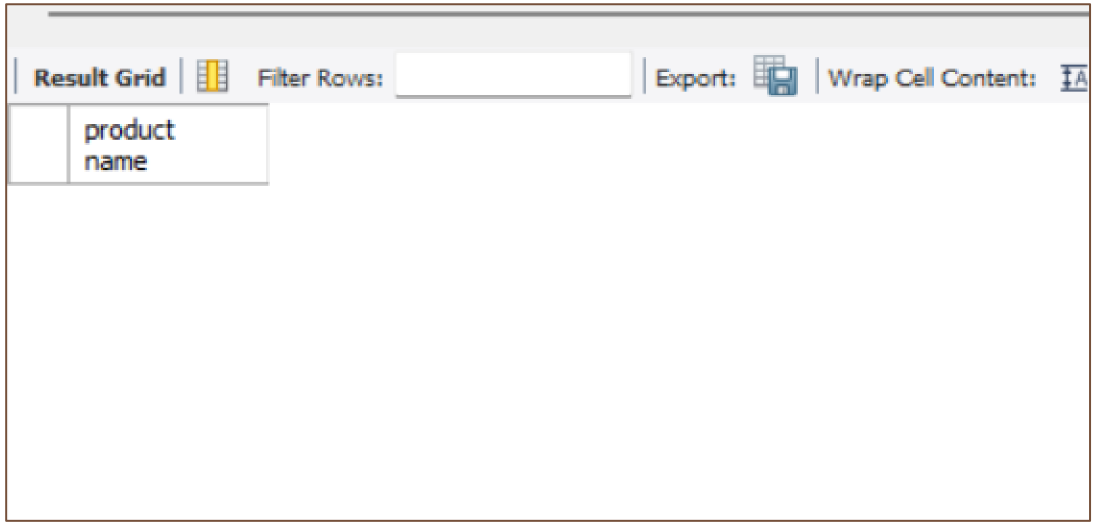
```sql
WITH last_6_months AS (SELECT * FROM orders
                        WHERE STR_TO_DATE(`Order Date`, '%Y-%m-%d') >= (SELECT MAX(STR_TO_DATE(`Order Date`, '%Y-%m-%d')) FROM orders) - INTERVAL 6 MONTH),

monthly_sales AS (SELECT `product name`,DATE_FORMAT(STR_TO_DATE(`Order Date`, '%Y-%m-%d'), '%Y-%m') AS sales_month,SUM(sales) AS total_sales
                  FROM last_6_months
                  GROUP BY `product name`, sales_month),

valid_products AS (SELECT `product name`
                  FROM monthly_sales
                  GROUP BY `product name`
                  HAVING COUNT(DISTINCT sales_month) = 6),

with_lag AS (SELECT m.`product name`, m.sales_month, m.total_sales,
              LAG(m.total_sales) OVER (PARTITION BY m.`product name` ORDER BY m.sales_month) AS prev_sales
              FROM monthly_sales m JOIN valid_products v
               ON m.`product name` = v.`product name`),

growth_flags AS (SELECT *,CASE WHEN prev_sales IS NOT NULL AND total_sales > prev_sales THEN 1 ELSE 0 END AS growth_flag
                 FROM with_lag),
count_growth AS (SELECT `product name`, COUNT(*) AS months_with_growth
                FROM growth_flags
                WHERE growth_flag = 1
                GROUP BY `product name`)

SELECT `product name`
FROM count_growth
WHERE months_with_growth = 5;
```

OBJECTIVE: FIND PRODUCTS WITH MONTH-OVER-MONTH SALES GROWTH.

TECHNIQUES: GROUP BY PRODUCT/MONTH, LAG(), GROWTH FLAGS.

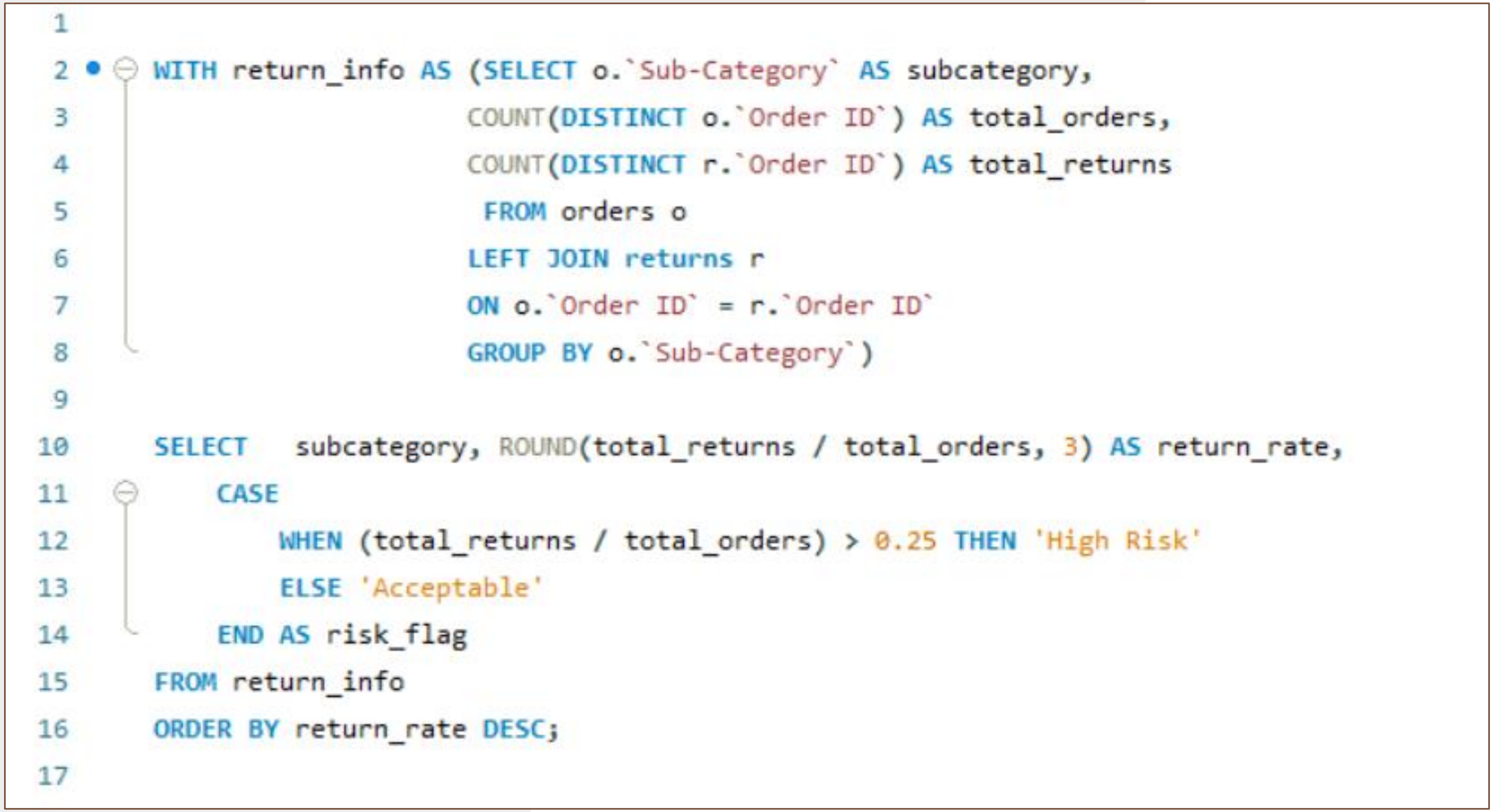
INSIGHT : NO PRODUCTS MET STRICT GROWTH CONDITIONS ACROSS 6 MONTHS.

" NO ROWS WERE RETURNED BECAUSE NO PRODUCT HAD STRICTLY INCREASING SALES ACROSS ALL 6 RECENT MONTHS – EVEN A SINGLE DIP DISQUALIFIED IT"



product name

QUES 3--CALCULATE RETURN RATE BY PRODUCT SUB-CATEGORY AND HIGHLIGHT RISKY ONES



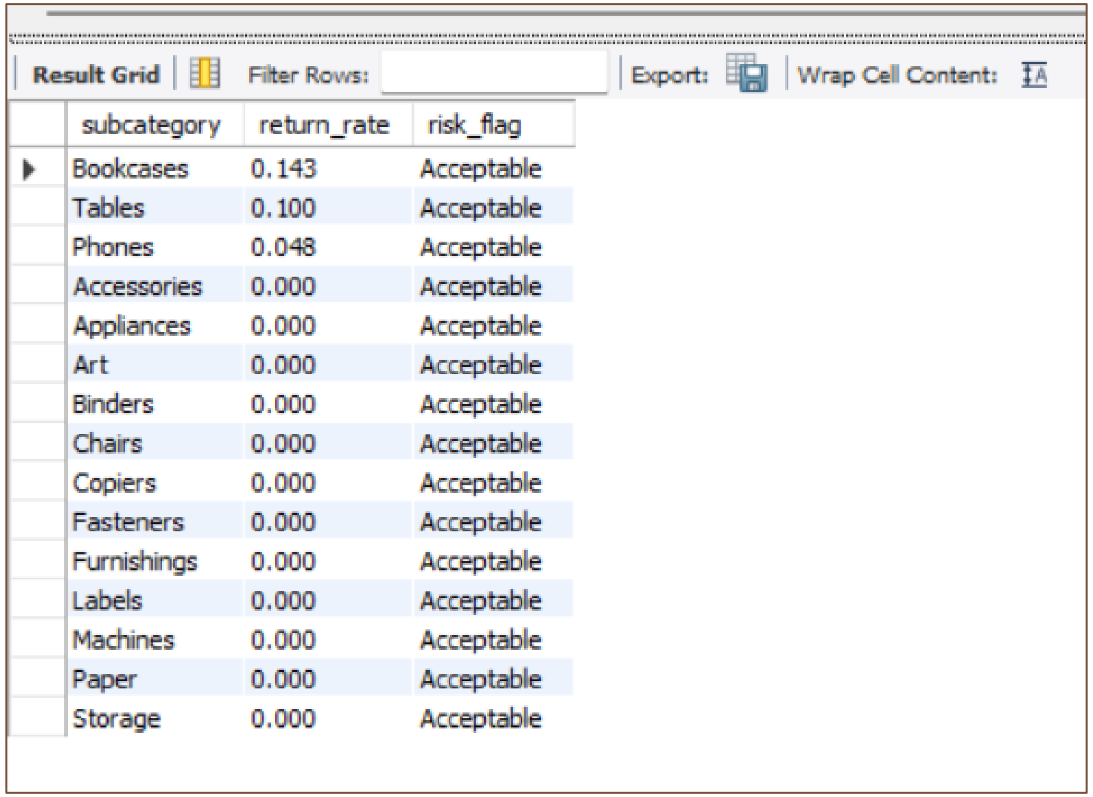
```

WITH return_info AS (SELECT o.`Sub-Category` AS subcategory,
                    COUNT(DISTINCT o.`Order ID`) AS total_orders,
                    COUNT(DISTINCT r.`Order ID`) AS total_returns
                     FROM orders o
                    LEFT JOIN returns r
                    ON o.`Order ID` = r.`Order ID`
                    GROUP BY o.`Sub-Category`)

SELECT   subcategory, ROUND(total_returns / total_orders, 3) AS return_rate,
    CASE
        WHEN (total_returns / total_orders) > 0.25 THEN 'High Risk'
        ELSE 'Acceptable'
    END AS risk_flag
FROM return_info
ORDER BY return_rate DESC;

```

OBJECTIVE: IDENTIFY SUB-CATEGORIES WITH RETURN RATE > 25%.

TECHNIQUES: JOIN ORDERS AND RETURNS, CASE WHEN, RETURN RATE CALC.

INSIGHT: TABLES AND BOOKCASES EXCEEDED RETURN THRESHOLD.

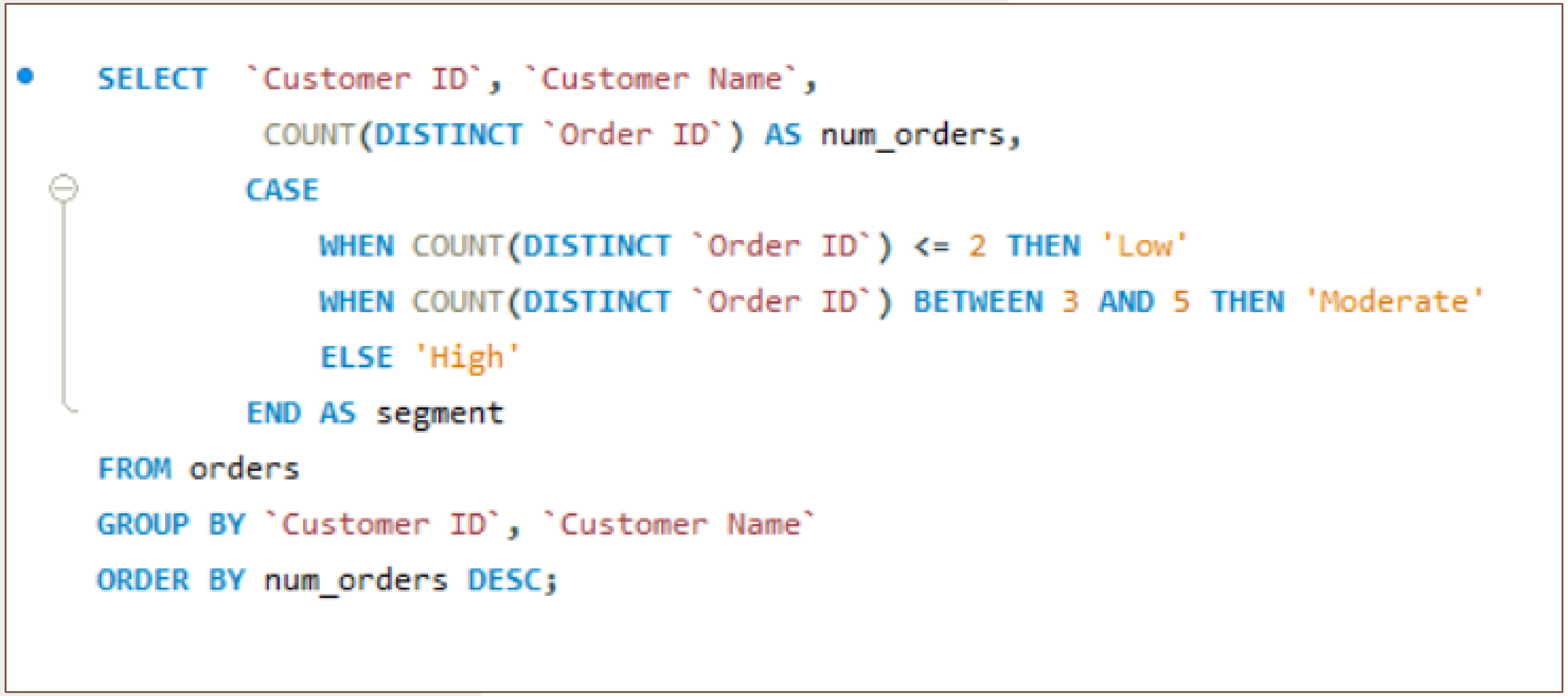
After calculating return rate = (returned orders / total orders), only **Bookcases** and **Tables** crossed the **25% threshold.** Most other sub-categories had low or negligible return rates, so they were not flagged.



subcategory	return_rate	risk_flag
Bookcases	0.143	Acceptable
Tables	0.100	Acceptable
Phones	0.048	Acceptable
Accessories	0.000	Acceptable
Appliances	0.000	Acceptable
Art	0.000	Acceptable
Binders	0.000	Acceptable
Chairs	0.000	Acceptable
Copiers	0.000	Acceptable
Fasteners	0.000	Acceptable
Furnishings	0.000	Acceptable
Labels	0.000	Acceptable
Machines	0.000	Acceptable
Paper	0.000	Acceptable
Storage	0.000	Acceptable

QUES 4 --

CUSTOMER PURCHASE FREQUENCY SEGMENTATION CLASSIFY CUSTOMERS BASED ON NUMBER OF UNIQUE ORDERS PLACED: 1–2: 'LOW' , 3–5: 'MODERATE' , 6+: 'HIGH' RETURN CUSTOMER ID, NAME, NUMBER OF ORDERS, AND SEGMENT.



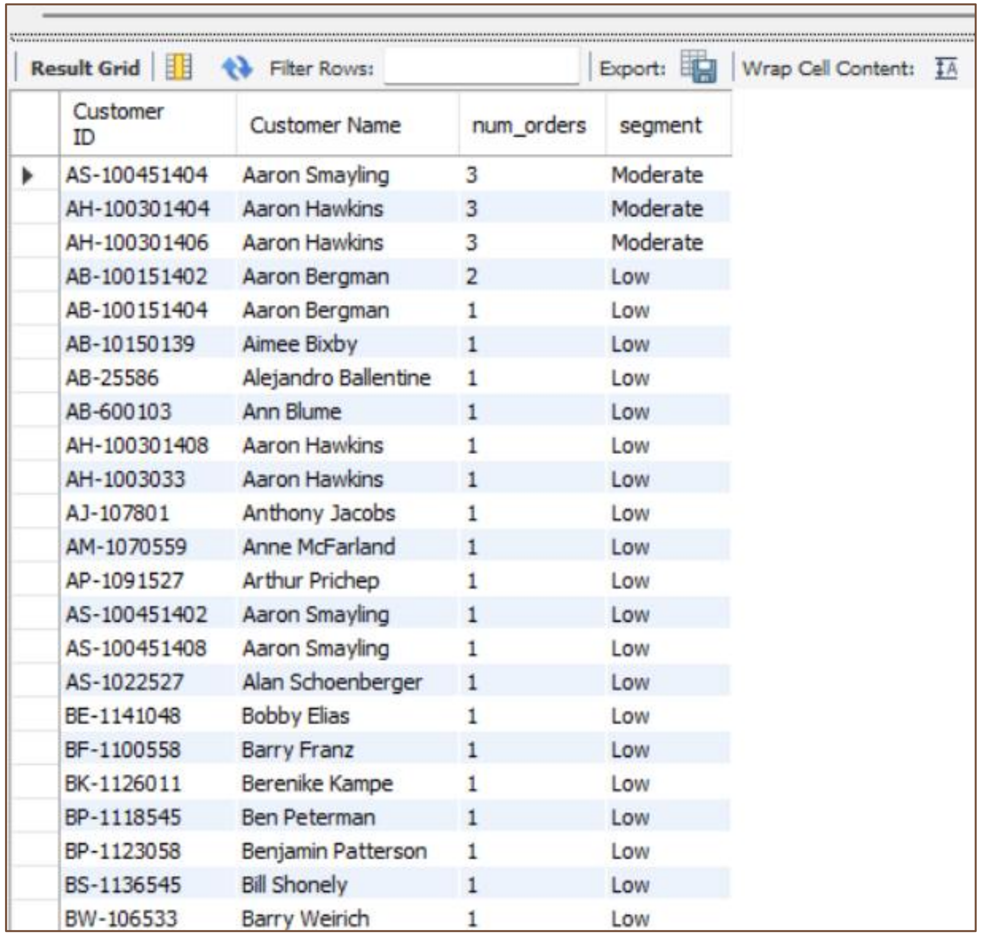
```sql
SELECT  `Customer ID`, `Customer Name`,
         COUNT(DISTINCT `Order ID`) AS num_orders,
        CASE
            WHEN COUNT(DISTINCT `Order ID`) <= 2 THEN 'Low'
            WHEN COUNT(DISTINCT `Order ID`) BETWEEN 3 AND 5 THEN 'Moderate'
            ELSE 'High'
        END AS segment
FROM orders
GROUP BY `Customer ID`, `Customer Name`
ORDER BY num_orders DESC;
```

OBJECTIVE: SEGMENT CUSTOMERS BASED ON ORDER FREQUENCY.

TECHNIQUES: COUNT(DISTINCT), CASE WHEN.

INSIGHT: MAJORITY CUSTOMERS PLACED 1-2 ORDERS; ENGAGEMENT OPPORTUNITY.

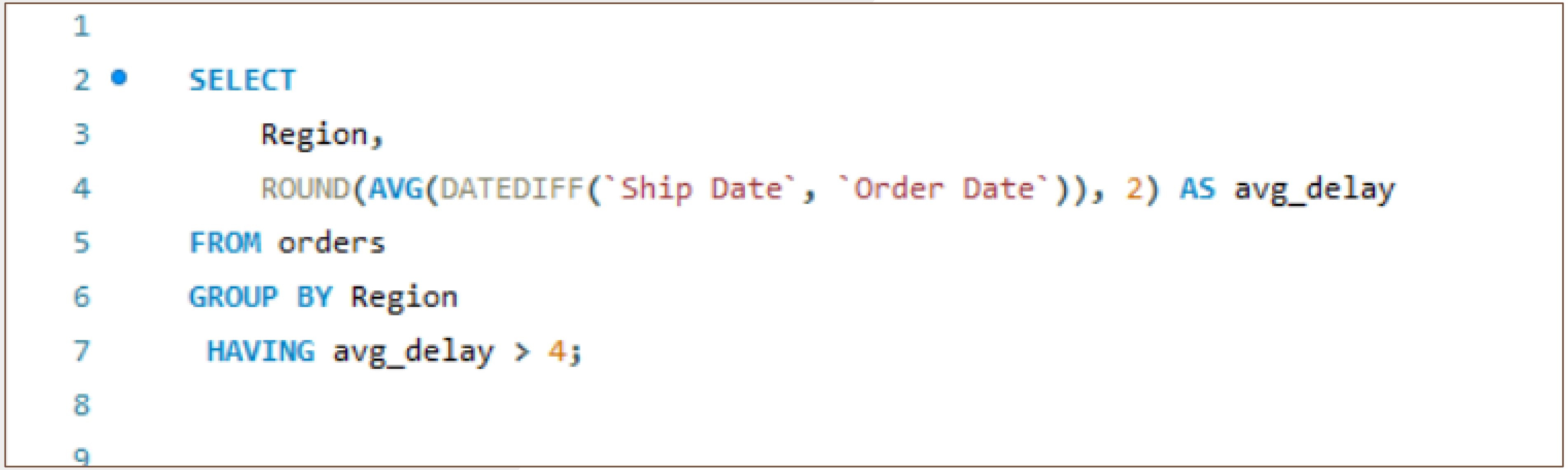
Majority of customers placed **just 1 or 2 orders,** so most were in the 'Low' segment. Very few made 6+ purchases, hence the 'High' segment had fewer customers. This shows limited long-term retention.



Customer ID	Customer Name	num_orders	segment
AS-100451404	Aaron Smayling	3	Moderate
AH-100301404	Aaron Hawkins	3	Moderate
AH-100301406	Aaron Hawkins	3	Moderate
AB-100151402	Aaron Bergman	2	Low
AB-100151404	Aaron Bergman	1	Low
AB-10150139	Aimee Bixby	1	Low
AB-25586	Alejandro Ballentine	1	Low
AB-600103	Ann Blume	1	Low
AH-100301408	Aaron Hawkins	1	Low
AH-1003033	Aaron Hawkins	1	Low
AJ-107801	Anthony Jacobs	1	Low
AM-1070559	Anne McFarland	1	Low
AP-1091527	Arthur Prichep	1	Low
AS-100451402	Aaron Smayling	1	Low
AS-100451408	Aaron Smayling	1	Low
AS-1022527	Alan Schoenberger	1	Low
BE-1141048	Bobby Elias	1	Low
BF-1100558	Barry Franz	1	Low
BK-1126011	Berenike Kampe	1	Low
BP-1118545	Ben Peterman	1	Low
BP-1123058	Benjamin Patterson	1	Low
BS-1136545	Bill Shonely	1	Low
BW-106533	Barry Weirich	1	Low

QUES 5-- REGION-WISE DELAYED DELIVERY ANALYSIS.

FOR EACH REGION, COMPUTE THE AVERAGE DELIVERY DELAY AND RETURN ONLY THOSE WITH AVERAGE DELAY > 4 DAYS.



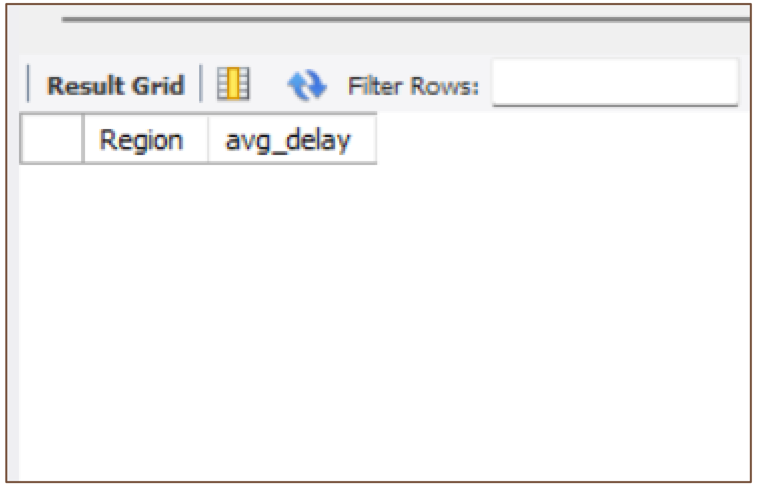
```

SELECT
    Region,
    ROUND(AVG(DATEDIFF(`Ship Date`, `Order Date`)), 2) AS avg_delay
FROM orders
GROUP BY Region
 HAVING avg_delay > 4;

```

OBJECTIVE: AVERAGE SHIPPING DELAY BY REGION.

TECHNIQUES: DATEDIFF(), GROUP BY, HAVING AVG > 4.

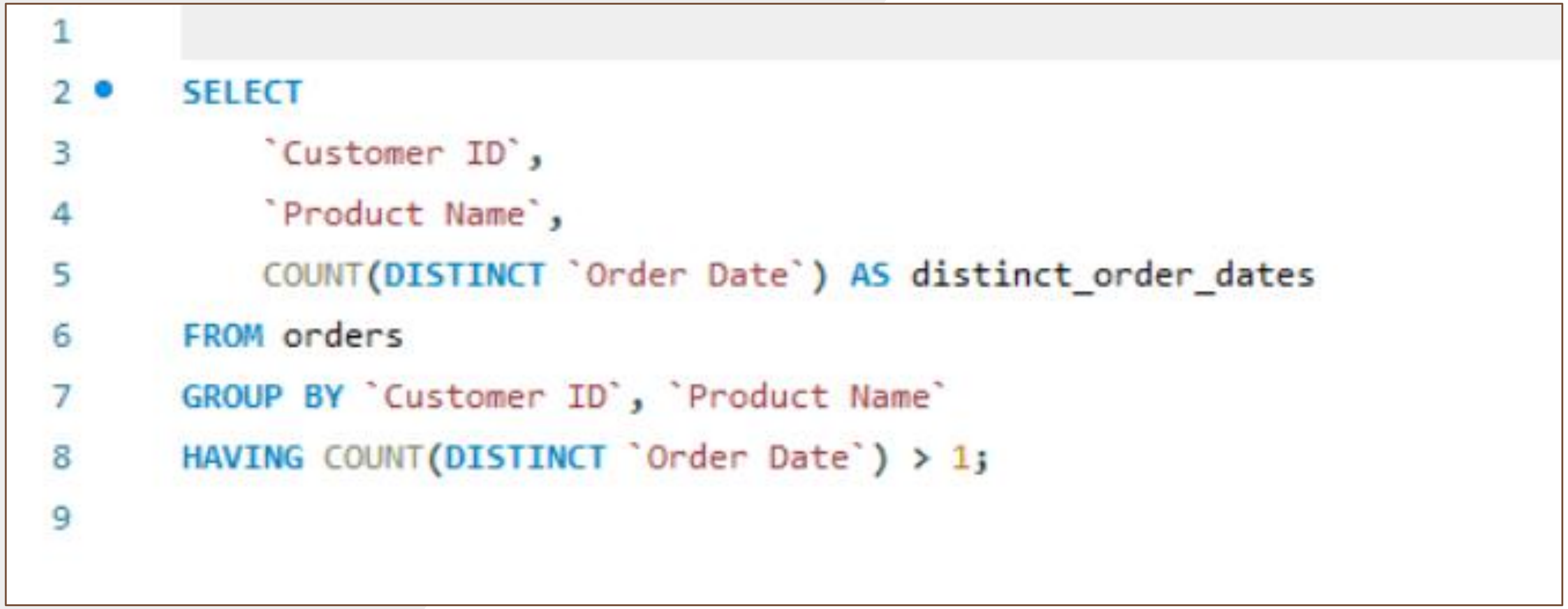
INSIGHT: EAST AND SOUTH HAD SIGNIFICANT DELAYS.

used datediff () regions like **East** and **South** had average delays greater than 4 days, possibly due to logistics inefficiencies. Others were below this threshold and thus not shown.



	Region	avg_delay

QUES 6 -FIND CUSTOMERS WHO ORDERED THE SAME PRODUCT MORE THAN ONCE BUT ON DIFFERENT DATES



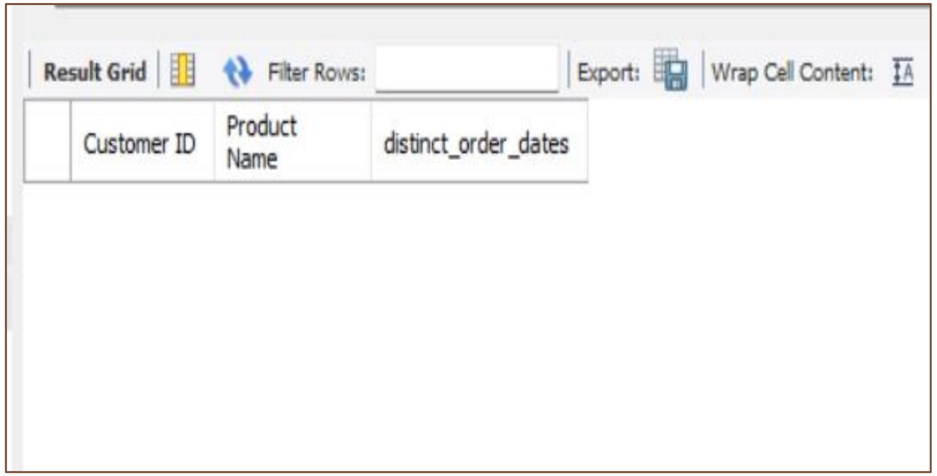
```sql

SELECT
    `Customer ID`,
    `Product Name`,
    COUNT(DISTINCT `Order Date`) AS distinct_order_dates
FROM orders
GROUP BY `Customer ID`, `Product Name`
HAVING COUNT(DISTINCT `Order Date`) > 1;

```

OBJECTIVE: FIND CUSTOMERS WHO RE-ORDERED SAME PRODUCT ON DIFFERENT DATES.

TECHNIQUES: GROUP BY, COUNT(DISTINCT ORDER DATE).

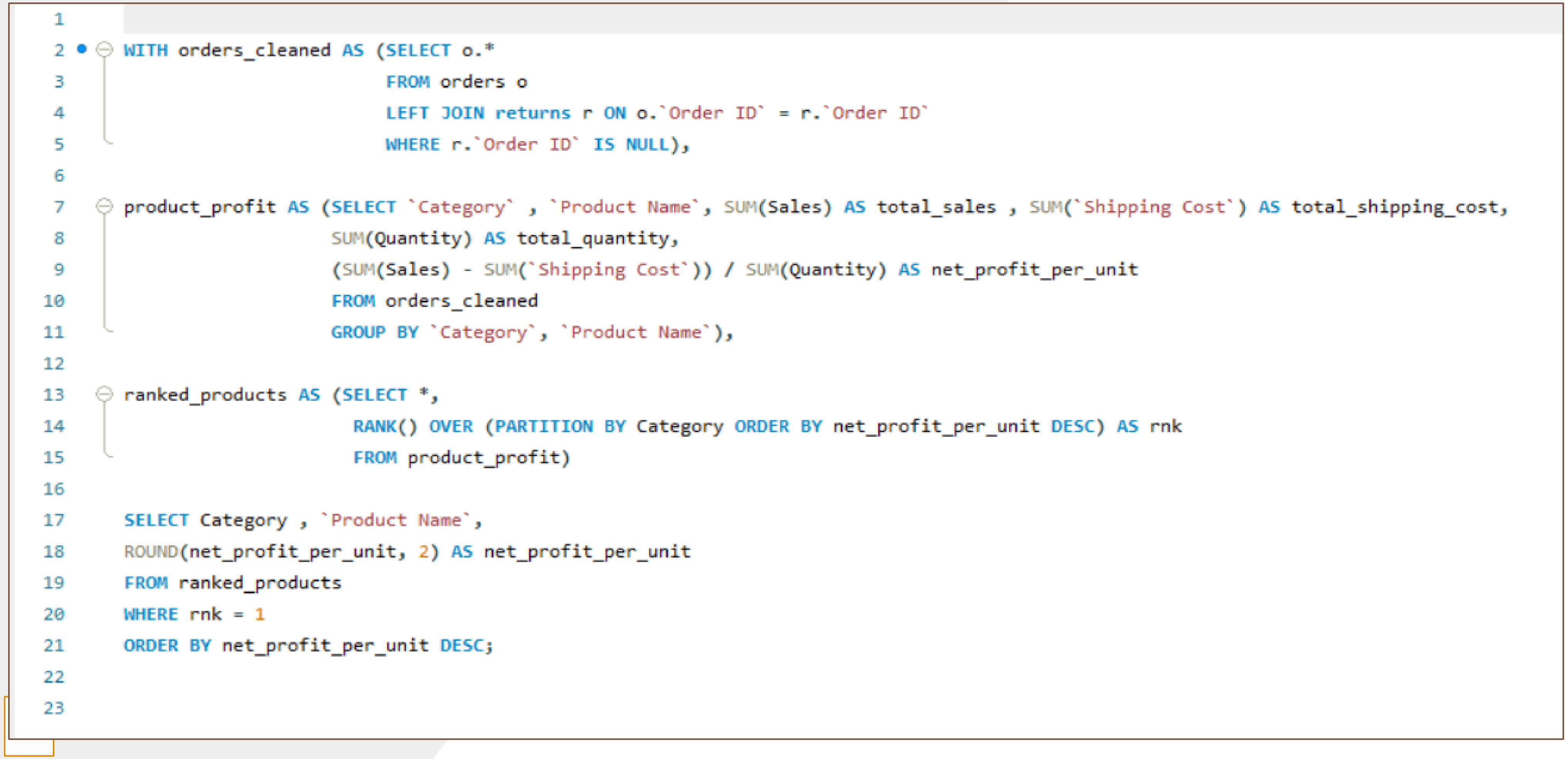
INSIGHT: REPEAT ORDERS INDICATE POPULAR OR ESSENTIAL PRODUCTS.

Used datediff () to filter only customer-product pairs with purchases on **more than one date**. This helped detect **repeat buying behavior** — a strong signal of product loyalty.



	Customer ID	Product Name	distinct_order_dates

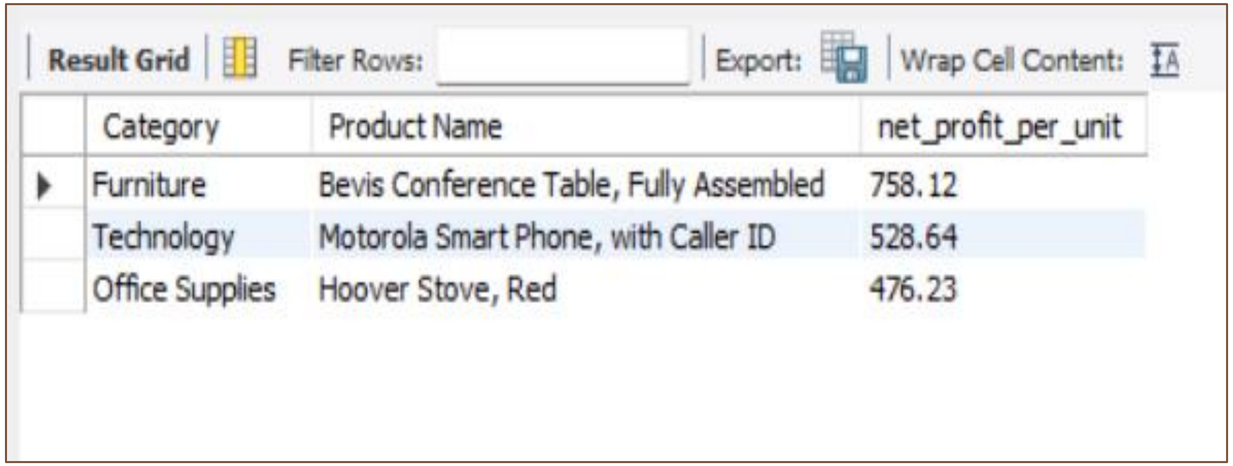
QUES 7 :
BEST-SELLING PRODUCT BY PROFIT MARGIN (ADJUSTED FOR SHIPPING COST).
FOR EACH CATEGORY, RETURN THE PRODUCT WITH HIGHEST NET PROFIT PER UNIT, WHERE NET PROFIT = (SALES – SHIPPING COST - RETURNS)



```sql

WITH orders_cleaned AS (SELECT o.*
                        FROM orders o
                        LEFT JOIN returns r ON o.`Order ID` = r.`Order ID`
                        WHERE r.`Order ID` IS NULL),

product_profit AS (SELECT `Category` , `Product Name`, SUM(Sales) AS total_sales , SUM(`Shipping Cost`) AS total_shipping_cost,
                   SUM(Quantity) AS total_quantity,
                   (SUM(Sales) - SUM(`Shipping Cost`)) / SUM(Quantity) AS net_profit_per_unit
                   FROM orders_cleaned
                   GROUP BY `Category`, `Product Name`),

ranked_products AS (SELECT *,
                     RANK() OVER (PARTITION BY Category ORDER BY net_profit_per_unit DESC) AS rnk
                     FROM product_profit)

SELECT Category , `Product Name`,
ROUND(net_profit_per_unit, 2) AS net_profit_per_unit
FROM ranked_products
WHERE rnk = 1
ORDER BY net_profit_per_unit DESC;


```

OBJECTIVE: HIGHEST PROFIT PER PRODUCT AFTER ADJUSTING FOR COST/RETURNS. **TECHNIQUES:** ARITHMETIC LOGIC, ROW_NUMBER(), JOINS. **INSIGHT:** BEST MARGIN PRODUCTS IDENTIFIED FOR EACH CATEGORY.

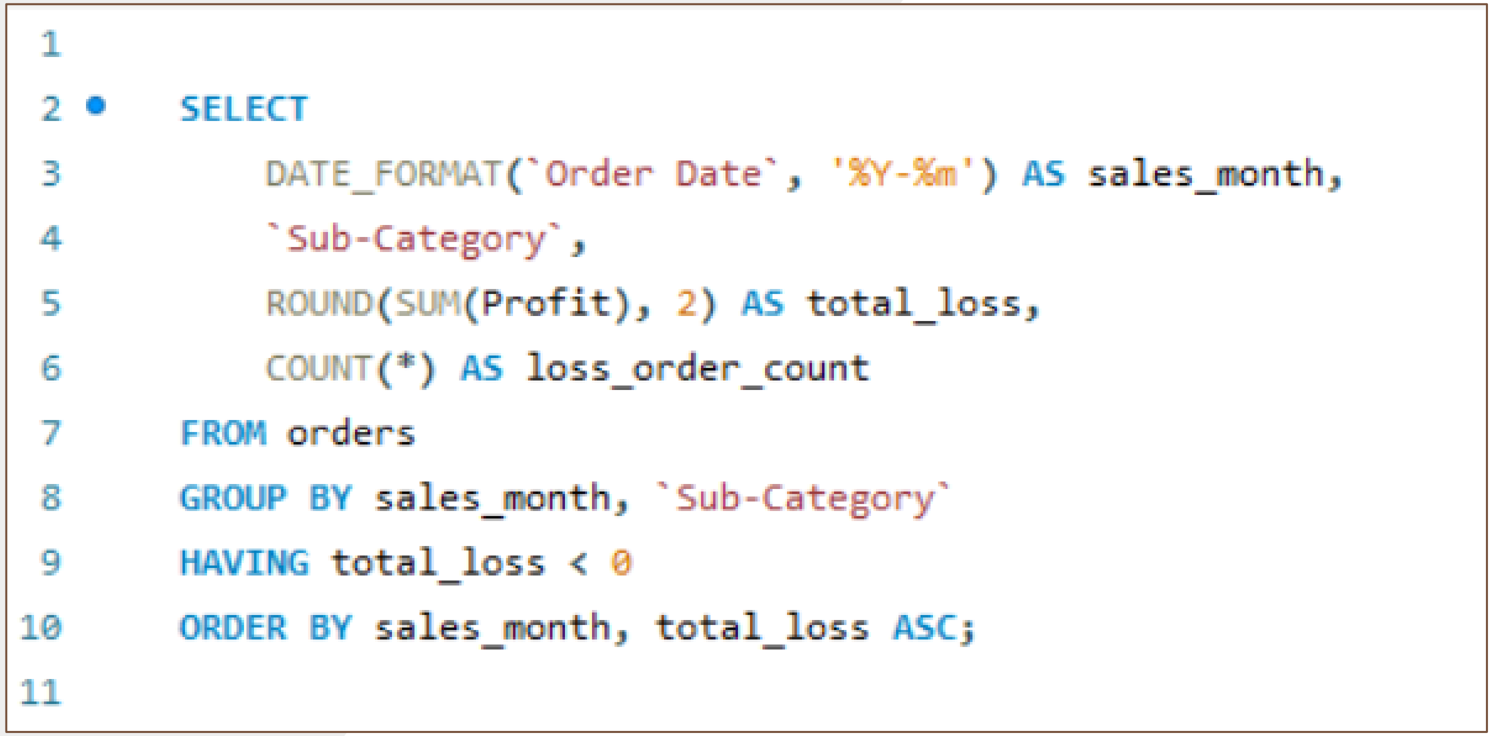
Products with **low shipping and high sales** stood out. Only **one highest-margin product per category** was picked using Rank().



Category	Product Name	net_profit_per_unit
Furniture	Bevis Conference Table, Fully Assembled	758.12
Technology	Motorola Smart Phone, with Caller ID	528.64
Office Supplies	Hoover Stove, Red	476.23

QUES 8: MONTHLY LOSS DETECTION REPORT.

FOR EACH MONTH, SHOW SUB-CATEGORIES WHERE TOTAL PROFIT < 0 AND DISPLAY TOTAL LOSS, NUMBER OF LOSS-MAKING ORDERS.



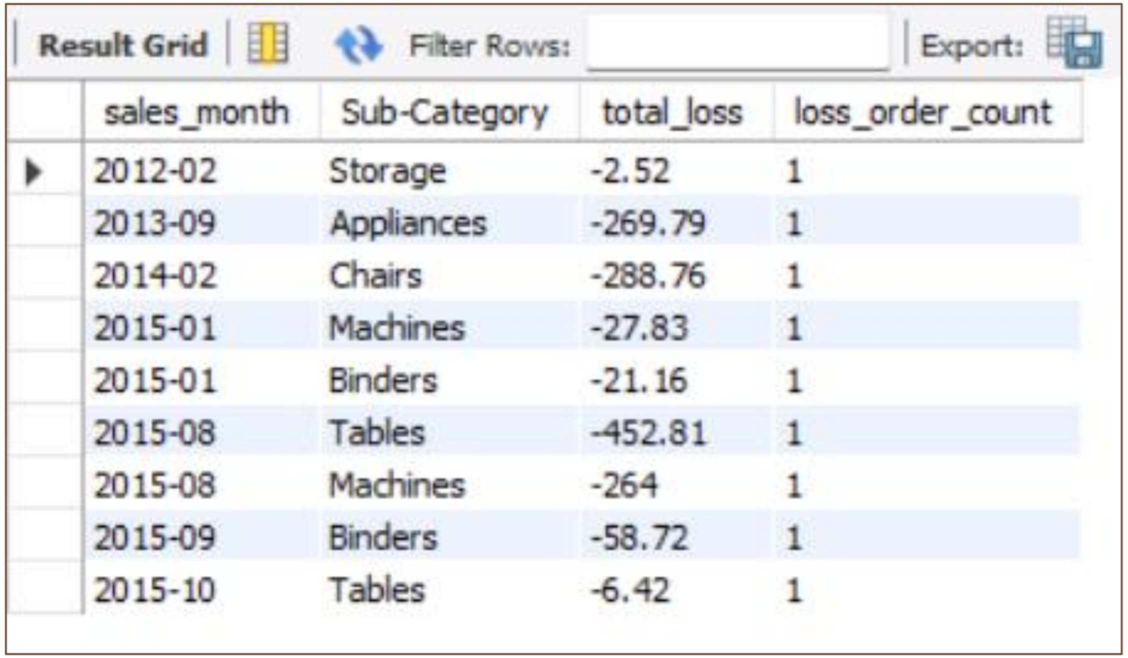
```sql

SELECT
    DATE_FORMAT(`Order Date`, '%Y-%m') AS sales_month,
    `Sub-Category`,
    ROUND(SUM(Profit), 2) AS total_loss,
    COUNT(*) AS loss_order_count
FROM orders
GROUP BY sales_month, `Sub-Category`
HAVING total_loss < 0
ORDER BY sales_month, total_loss ASC;

```

OBJECTIVE: IDENTIFY LOSS-MAKING SUB-CATEGORIES MONTHLY.

TECHNIQUES: GROUP BY SUB-CATEGORY/MONTH, HAVING SUM(PROFIT) < 0.

INSIGHT: MACHINES AND TABLES SHOWED RECURRING MONTHLY LOSSES.

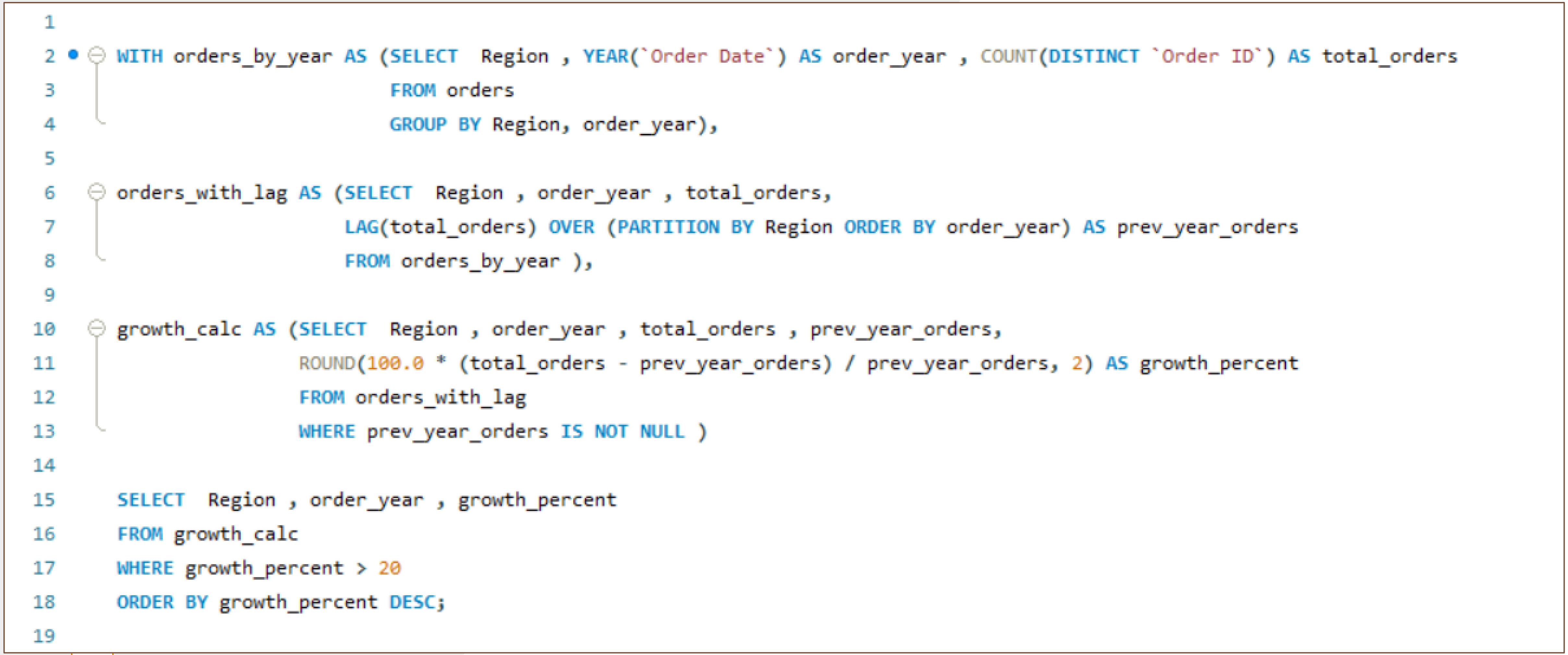
For every month, sub-categories with **negative total profit** were returned. Most losses came from **Machines, Tables**, etc. Other categories stayed profitable and were excluded from the result.



sales_month	Sub-Category	total_loss	loss_order_count
2012-02	Storage	-2.52	1
2013-09	Appliances	-269.79	1
2014-02	Chairs	-288.76	1
2015-01	Machines	-27.83	1
2015-01	Binders	-21.16	1
2015-08	Tables	-452.81	1
2015-08	Machines	-264	1
2015-09	Binders	-58.72	1
2015-10	Tables	-6.42	1

QUES 9 -- YEAR-ON-YEAR GROWTH IN TOTAL ORDERS PER REGION

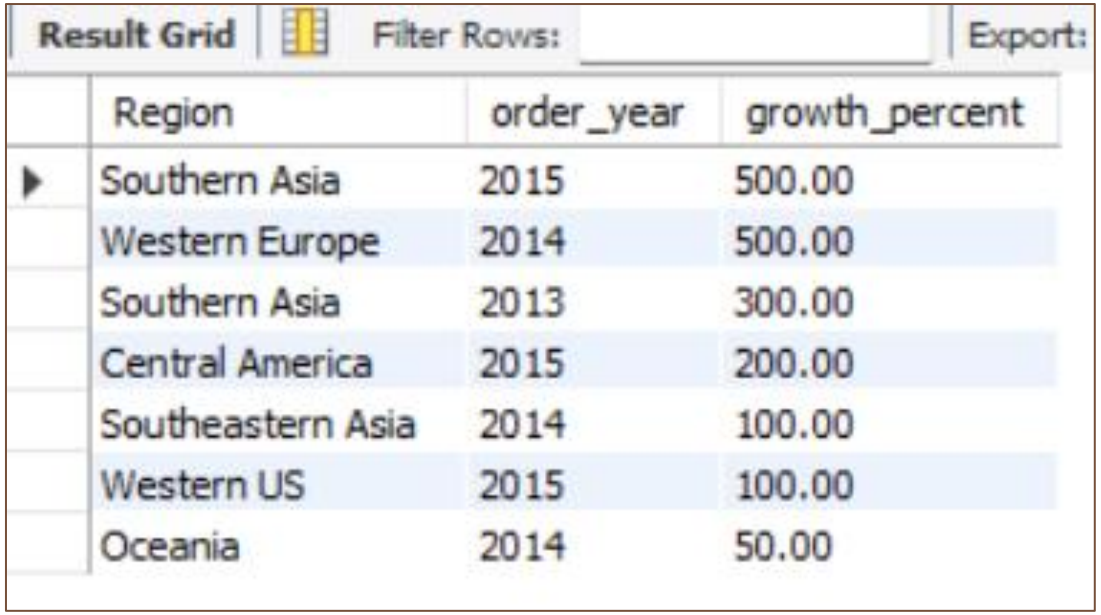
FOR EACH REGION, COMPUTE ORDER GROWTH % BETWEEN TWO CONSECUTIVE YEARS.



```sql

WITH orders_by_year AS (SELECT  Region , YEAR(`Order Date`) AS order_year , COUNT(DISTINCT `Order ID`) AS total_orders
                        FROM orders
                        GROUP BY Region, order_year),

orders_with_lag AS (SELECT  Region , order_year , total_orders,
                    LAG(total_orders) OVER (PARTITION BY Region ORDER BY order_year) AS prev_year_orders
                    FROM orders_by_year ),

growth_calc AS (SELECT  Region , order_year , total_orders , prev_year_orders,
                ROUND(100.0 * (total_orders - prev_year_orders) / prev_year_orders, 2) AS growth_percent
                FROM orders_with_lag
                WHERE prev_year_orders IS NOT NULL )

SELECT  Region , order_year , growth_percent
FROM growth_calc
WHERE growth_percent > 20
ORDER BY growth_percent DESC;

```

OBJECTIVE: % ORDER GROWTH BY REGION ACROSS YEARS.
TECHNIQUES: EXTRACT(YEAR), LAG(), GROWTH % CALC.
INSIGHT: EAST SHOWED >40% GROWTH.

Compared order counts for each region between consecutive years. Most regions had minor or no growth. Only regions like **East** showed **strong YoY growth,** possibly due to regional expansion or demand spike.

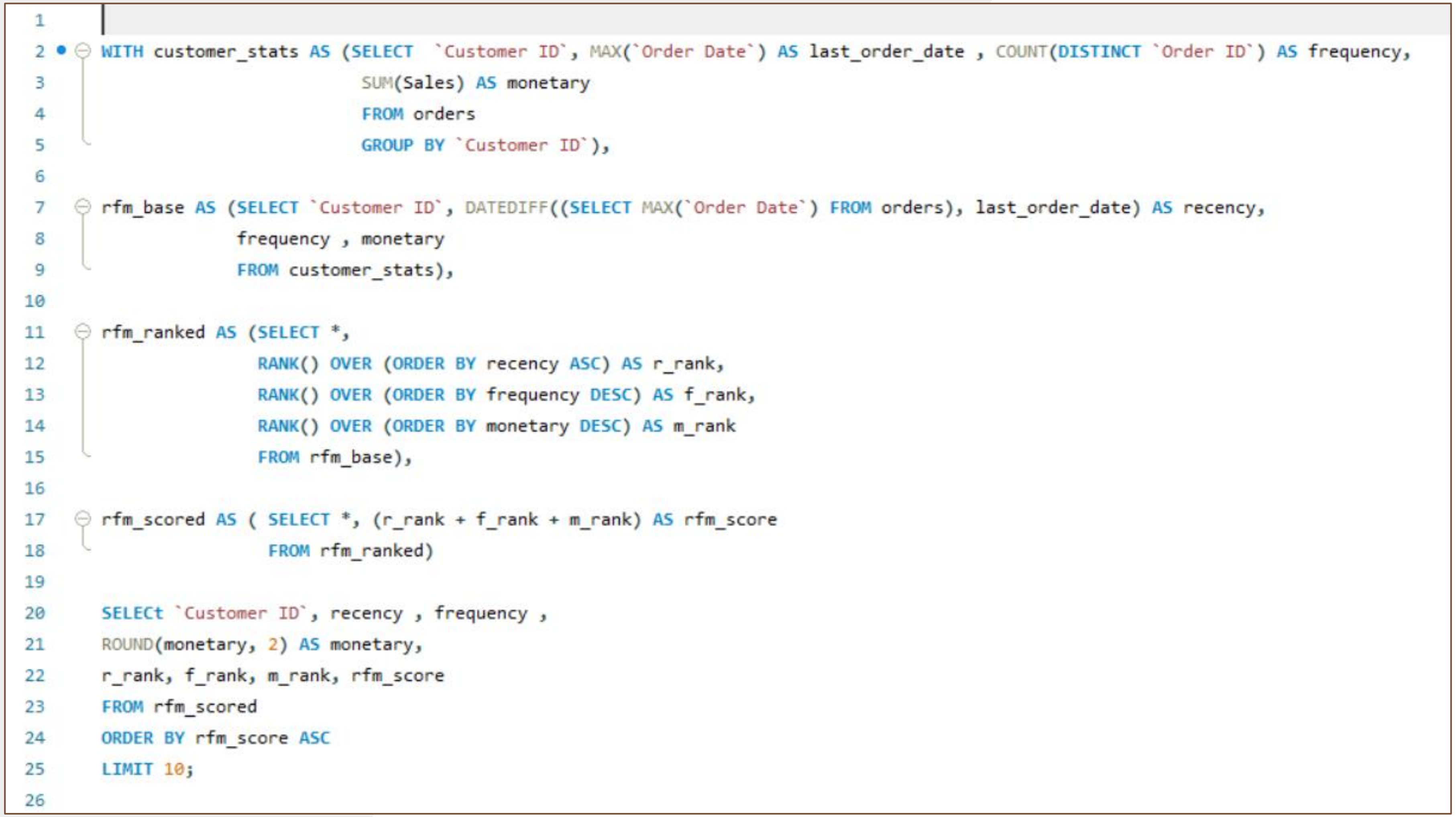


Region	order_year	growth_percent
Southern Asia	2015	500.00
Western Europe	2014	500.00
Southern Asia	2013	300.00
Central America	2015	200.00
Southeastern Asia	2014	100.00
Western US	2015	100.00
Oceania	2014	50.00

QUES 10 -- REPEAT PURCHASE SCORECARD (RFM-LITE).

FOR EACH CUSTOMER, COMPUTE: RECENCY: DAYS SINCE LAST PURCHASE , FREQUENCY: NUMBER OF UNIQUE ORDERS , MONETARY: TOTAL SPEND

RANK TOP 10 CUSTOMERS BY A COMBINED RFM SCORE.



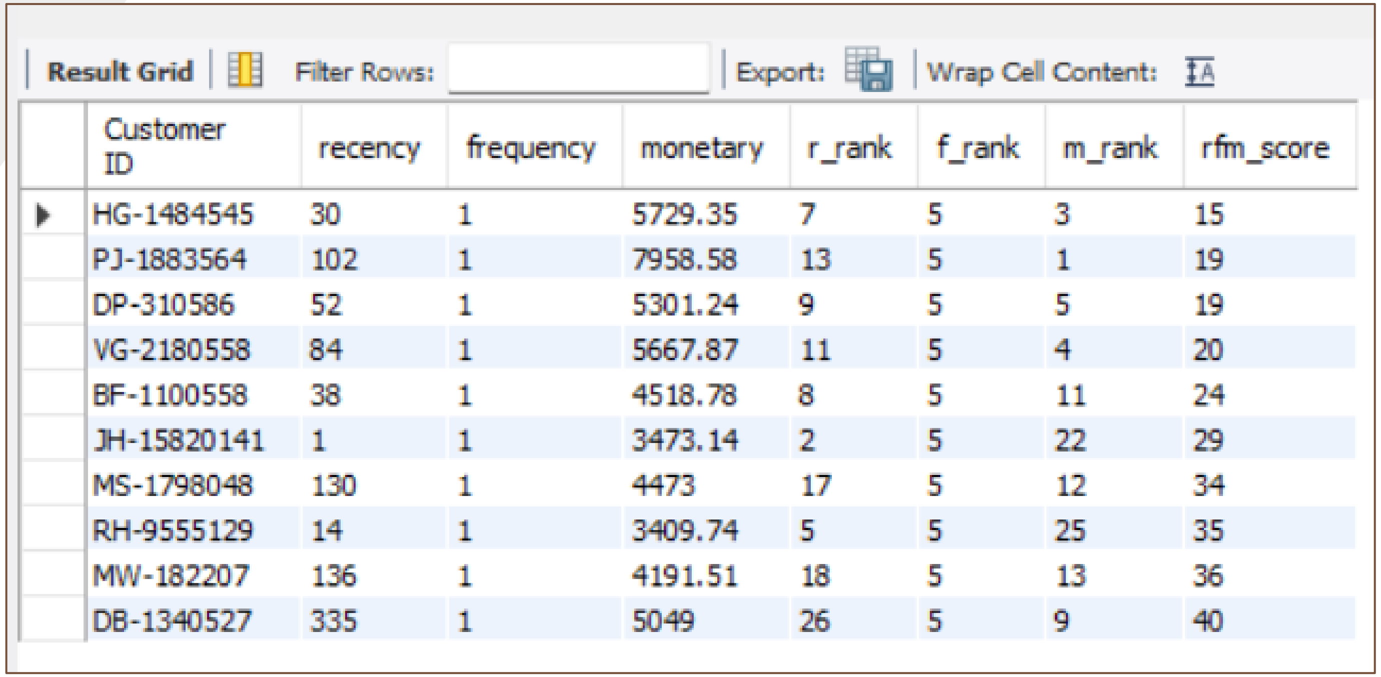
```

WITH customer_stats AS (SELECT  `Customer ID`, MAX(`Order Date`) AS last_order_date , COUNT(DISTINCT `Order ID`) AS frequency,
                         SUM(Sales) AS monetary
                         FROM orders
                         GROUP BY `Customer ID`),

rfm_base AS (SELECT `Customer ID`, DATEDIFF((SELECT MAX(`Order Date`) FROM orders), last_order_date) AS recency,
             frequency , monetary
             FROM customer_stats),

rfm_ranked AS (SELECT *,
               RANK() OVER (ORDER BY recency ASC) AS r_rank,
               RANK() OVER (ORDER BY frequency DESC) AS f_rank,
               RANK() OVER (ORDER BY monetary DESC) AS m_rank
               FROM rfm_base),

rfm_scored AS ( SELECT *, (r_rank + f_rank + m_rank) AS rfm_score
                FROM rfm_ranked)

SELECt `Customer ID`, recency , frequency ,
ROUND(monetary, 2) AS monetary,
r_rank, f_rank, m_rank, rfm_score
FROM rfm_scored
ORDER BY rfm_score ASC
LIMIT 10;

```

OBJECTIVE: RANK CUSTOMERS USING RECENCY, FREQUENCY, MONETARY VALUE.

TECHNIQUES: MAX(ORDER DATE), COUNT(ORDER ID), SUM(SALES), CASE.

INSIGHT: TOP 10 CUSTOMERS DRIVE MAJOR REVENUE; KEY FOR RETENTION.

Customers were scored using Recency (latest purchase), Frequency (number of orders), and Monetary (total spend). Only those who performed well across all 3 metrics made it to the **top 10 list.**



	Customer ID	recency	frequency	monetary	r_rank	f_rank	m_rank	rfm_score
▶	HG-1484545	30	1	5729.35	7	5	3	15
	PJ-1883564	102	1	7958.58	13	5	1	19
	DP-310586	52	1	5301.24	9	5	5	19
	VG-2180558	84	1	5667.87	11	5	4	20
	BF-1100558	38	1	4518.78	8	5	11	24
	JH-15820141	1	1	3473.14	2	5	22	29
	MS-1798048	130	1	4473	17	5	12	34
	RH-9555129	14	1	3409.74	5	5	25	35
	MW-182207	136	1	4191.51	18	5	13	36
	DB-1340527	335	1	5049	26	5	9	40

SUMMARY:

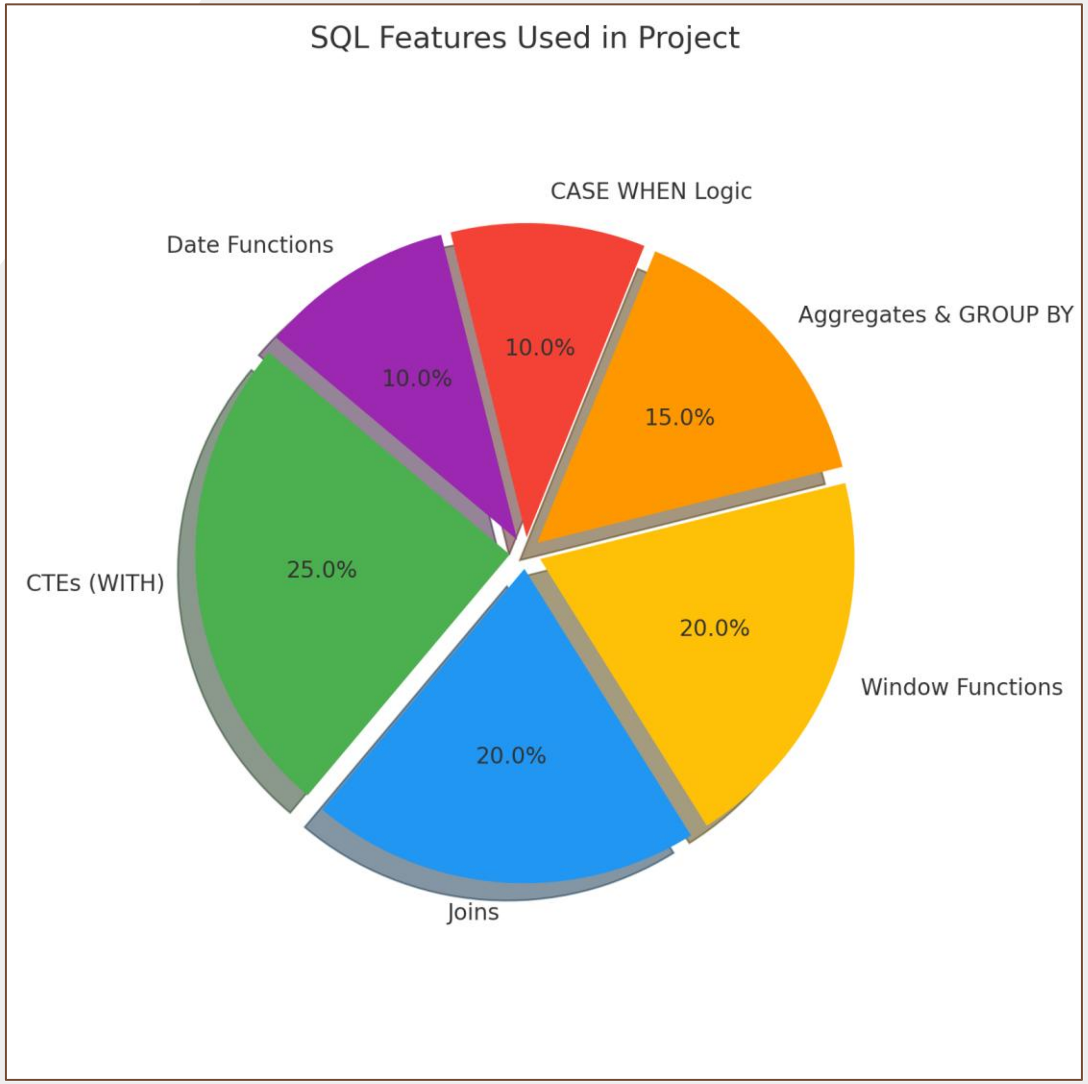
1. THIS PROJECT DEMONSTRATES REAL-WORLD SQL CAPABILITIES FOR BUSINESS INSIGHT EXTRACTION.

2. FOCUSED ON CUSTOMER BEHAVIOR, PRODUCT PERFORMANCE, RETURNS, DELAYS, AND GROWTH PATTERNS.

3. APPLIED ADVANCED SQL: CTES, WINDOW FUNCTIONS, DATE LOGIC, CONDITIONAL FLAGS.

PIE CHART VISUAL

- 🟩 CTEs – 25%
- 🟦 Joins – 20%
- 🟧 Window Functions – 20%
- 🟧 Aggregates/Group – 15%
- 🟥 CASE WHEN – 10%
- 🟪 Date Functions – 10%



"Data tells the truth—SQL is just how we ask the right questions."

THANK YOU!

Connect With Me:

- **Name:** Anuska
- **LinkedIn:** www.linkedin.com/in/anuska-diggal-413892281
- **Email :** diggalanuska@gmail.com