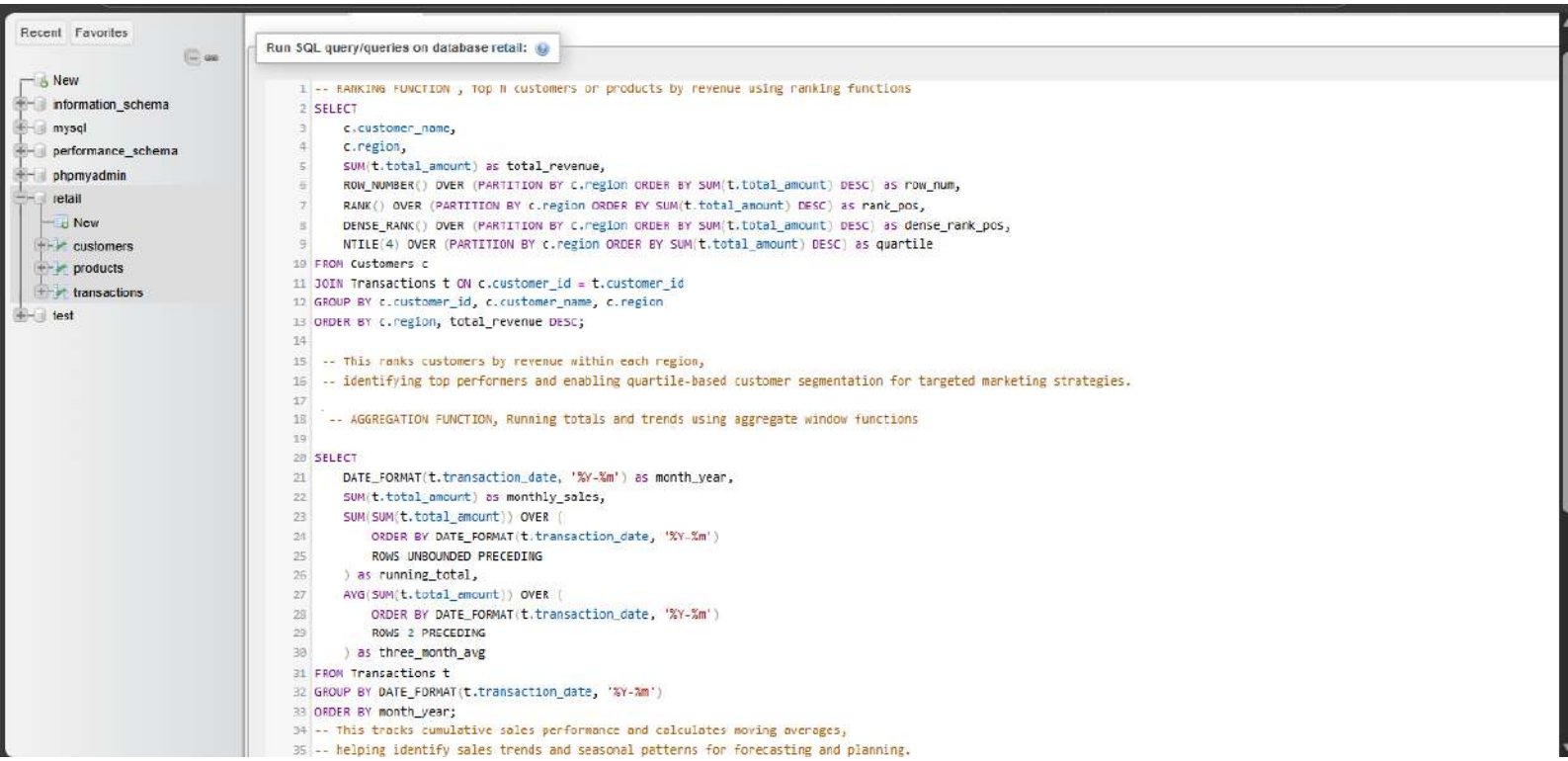
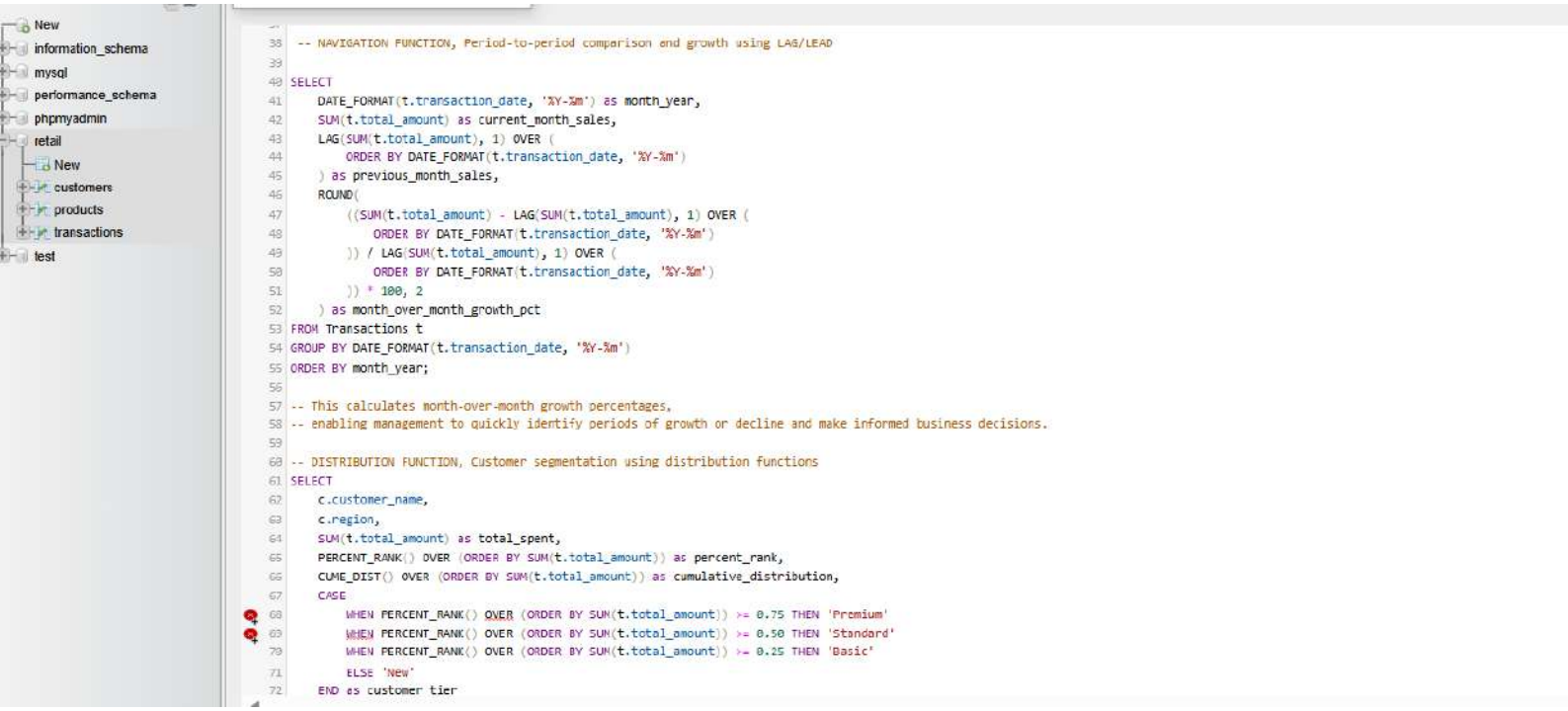

Recent
Favorites
New
information_schema
mysql
performance_schema
phpmyadmin
retail
New
customers
products
transactions
test
Run SQL query/queries on database retail:
-- RANKING FUNCTION , Top N customers or products by revenue using ranking functions
SELECT
c.customer_name,
c.region,
SUM(t.total_amount) as total_revenue,
ROW_NUMBER() OVER (PARTITION BY c.region ORDER BY SUM(t.total_amount) DESC) as row_num,
RANK() OVER (PARTITION BY c.region ORDER BY SUM(t.total_amount) DESC) as rank_pos,
DENSE_RANK() OVER (PARTITION BY c.region ORDER BY SUM(t.total_amount) DESC) as dense_rank_pos,
NTILE(4) OVER (PARTITION BY c.region ORDER BY SUM(t.total_amount) DESC) as quartile
FROM Customers c
JOIN Transactions t ON c.customer_id = t.customer_id
GROUP BY c.customer_id, c.customer_name, c.region
ORDER BY c.region, total_revenue DESC;
-- This ranks customers by revenue within each region,
-- identifying top performers and enabling quartile-based customer segmentation for targeted marketing strategies.
-- AGGREGATION FUNCTION, Running totals and trends using aggregate window functions
SELECT
DATE_FORMAT(t.transaction_date, '%Y-%m') as month_year,
SUM(t.total_amount) as monthly_sales,
SUM(SUM(t.total_amount)) OVER (
ORDER BY DATE_FORMAT(t.transaction_date, '%Y-%m')
ROWS UNBOUNDED PRECEDING
) as running_total,
AVG(SUM(t.total_amount)) OVER (
ORDER BY DATE_FORMAT(t.transaction_date, '%Y-%m')
ROWS 2 PRECEDING
) as three_month_avg
FROM Transactions t
GROUP BY DATE_FORMAT(t.transaction_date, '%Y-%m')
ORDER BY month_year;
-- This tracks cumulative sales performance and calculates moving averages,
-- helping identify sales trends and seasonal patterns for forecasting and planning.


New
information_schema
mysql
performance_schema
phpmyadmin
retail
New
customers
products
transactions
test
-- NAVIGATION FUNCTION, Period-to-period comparison and growth using LAG/LEAD
SELECT
DATE_FORMAT(t.transaction_date, '%Y-%m') as month_year,
SUM(t.total_amount) as current_month_sales,
LAG(SUM(t.total_amount), 1) OVER (
ORDER BY DATE_FORMAT(t.transaction_date, '%Y-%m')
) as previous_month_sales,
ROUND(
((SUM(t.total_amount) - LAG(SUM(t.total_amount), 1) OVER (
ORDER BY DATE_FORMAT(t.transaction_date, '%Y-%m')
)) / LAG(SUM(t.total_amount), 1) OVER (
ORDER BY DATE_FORMAT(t.transaction_date, '%Y-%m')
)) * 100, 2
) as month_over_month_growth_pct
FROM Transactions t
GROUP BY DATE_FORMAT(t.transaction_date, '%Y-%m')
ORDER BY month_year;
-- This calculates month-over-month growth percentages,
-- enabling management to quickly identify periods of growth or decline and make informed business decisions.
-- DISTRIBUTION FUNCTION, Customer segmentation using distribution functions
SELECT
c.customer_name,
c.region,
SUM(t.total_amount) as total_spent,
PERCENT_RANK() OVER (ORDER BY SUM(t.total_amount)) as percent_rank,
CUME_DIST() OVER (ORDER BY SUM(t.total_amount)) as cumulative_distribution,
CASE
WHEN PERCENT_RANK() OVER (ORDER BY SUM(t.total_amount)) >= 0.75 THEN 'Premium'
WHEN PERCENT_RANK() OVER (ORDER BY SUM(t.total_amount)) >= 0.50 THEN 'Standard'
WHEN PERCENT_RANK() OVER (ORDER BY SUM(t.total_amount)) >= 0.25 THEN 'Basic'
ELSE 'New'
END as customer_tier