# A Comprehensive Exercise on Grouping, Filtering, and Analyzing Data in MySQL #50

Part 1

Task 1: Product Performance Analysis

Query:

SELECT

p.productLine,

COUNT(od.productCode) AS totalProductsSold,

SUM(od.quantityOrdered \* od.priceEach) AS totalSalesAmount

FROM

products p, orderdetails od

WHERE

p.productCode = od.productCode

GROUP BY

p.productLine;

Explanation:

* This query counts the total number of products sold and calculates the total sales amount for each product line.
* Assumes a simple JOIN between the products and orderdetails tables based on the product code.
* Provides insights into the performance of each product line in terms of sales.

Task 2: Office Sales Analysis

Query:

SELECT

o.city AS officeCity,

COUNT(DISTINCT ord.orderNumber) AS numberOfOrdersProcessed,

SUM(odr.quantityOrdered \* odr.priceEach) AS totalSalesAmount

FROM

offices o

JOIN

employees e ON o.officeCode = e.officeCode

JOIN

customers c ON e.employeeNumber = c.salesRepEmployeeNumber

JOIN

orders ord ON c.customerNumber = ord.customerNumber

JOIN

orderdetails odr ON ord.orderNumber = odr.orderNumber

GROUP BY

o.city;

Explanation:

* This query calculates the number of orders processed and the total sales amount for each office city.
* Involves multiple JOIN operations to link offices, employees, customers, orders, and order details.
* Offers insights into the sales performance of each office, considering the number of orders and total sales amount.

Part 2

Task 1: High-Value Order Analysis

Query:

SELECT

o.city AS officeCity,

AVG(odr.quantityOrdered \* odr.priceEach) AS averageOrderValue,

COUNT(DISTINCT ord.orderNumber) AS totalOrders

FROM

offices o

JOIN

employees e ON o.officeCode = e.officeCode

JOIN

customers c ON e.employeeNumber = c.salesRepEmployeeNumber

JOIN

orders ord ON c.customerNumber = ord.customerNumber

JOIN

orderdetails odr ON ord.orderNumber = odr.orderNumber

GROUP BY

o.city

HAVING

averageOrderValue > 100;

Explanation:

* This query calculates the average order value for each office city and filters out offices with an average order value above $100.
* Involves similar JOIN operations as Task 2 in Part 1.
* Identifies offices with potentially high-value orders.

Task 2: Product Line Performance Filter

Query:

SELECT

p.productLine,

AVG(p.buyPrice) AS averageProductSalePrice

FROM

products p

GROUP BY

p.productLine

HAVING

averageProductSalePrice > 40;

Explanation:

* This query calculates the average sale price for each product line and filters out product lines with an average sale price above $40.
* Provides insights into product lines with higher-than-average sale prices.

Part 3

Task 1: Average Population and Total GDP by Continent

Query:

use world;

SELECT

Continent,

AVG(Population) AS AveragePopulation,

SUM(GNP) AS TotalGDP

FROM

country

GROUP BY

Continent

HAVING

AveragePopulation >= 5000000;

Explanation:

* This query calculates the average population and total GDP for each continent.
* Filters out continents with an average population below 5 million.

Task 2: Countries with Multiple Official Languages

Query:

SELECT

c.Name AS CountryName,

COUNT(cl.Language) AS NumberOfLanguages,

SUM(c.Population) AS TotalPopulation

FROM

country c

JOIN

countrylanguage cl ON c.Code = cl.CountryCode

GROUP BY

c.Code, c.Name

HAVING

NumberOfLanguages > 4;

Explanation:

* Identifies countries with more than four official languages, displaying the country name, the number of official languages, and total population.
* Considers language diversity in countries.

Part 4

Task 1: Monthly Sales Growth

Query:

USE classicmodels;

WITH MonthlySales AS (

SELECT

DATE\_FORMAT(o.orderDate, '%Y-%m') AS month,

p.productLine,

SUM(od.quantityOrdered \* od.priceEach) AS monthlySales

FROM

orders o

JOIN

orderdetails od ON o.orderNumber = od.orderNumber

JOIN

products p ON od.productCode = p.productCode

GROUP BY

month, p.productLine

)

SELECT

month,

productLine,

monthlySales,

(monthlySales / LAG(monthlySales) OVER (PARTITION BY productLine ORDER BY month) - 1) \* 100 AS salesGrowth

FROM

MonthlySales;

Explanation:

* This query calculates month-over-month sales growth percentage for each product line.
* Utilizes a Common Table Expression (CTE) to compute monthly sales.
* The LAG window function compares current month sales to the previous month for each product line.

Task 2: Quarterly Sales Analysis

Query:

USE classicmodels;

WITH QuarterlySales AS (

SELECT

e.officeCode,

CONCAT(YEAR(o.orderDate), 'Q', QUARTER(o.orderDate)) AS quarter,

SUM(od.quantityOrdered \* od.priceEach) AS quarterlySales

FROM

orders o

JOIN

orderdetails od ON o.orderNumber = od.orderNumber

JOIN

customers c ON o.customerNumber = c.customerNumber

JOIN

employees e ON c.salesRepEmployeeNumber = e.employeeNumber

GROUP BY

e.officeCode, quarter

)

SELECT

qs.officeCode,

qs.quarter,

qs.quarterlySales,

(qs.quarterlySales / AVG(qs.quarterlySales) OVER (PARTITION BY qs.officeCode)) \* 100 AS salesPercentage

FROM

QuarterlySales qs;

Explanation:

* This query performs quarterly sales analysis, calculating sales percentage compared to the average for each office.
* Utilizes a Common Table Expression (CTE) to calculate quarterly sales.
* The query then calculates the sales percentage using the AVG window function.