CLACGIC MODEL DATABSE

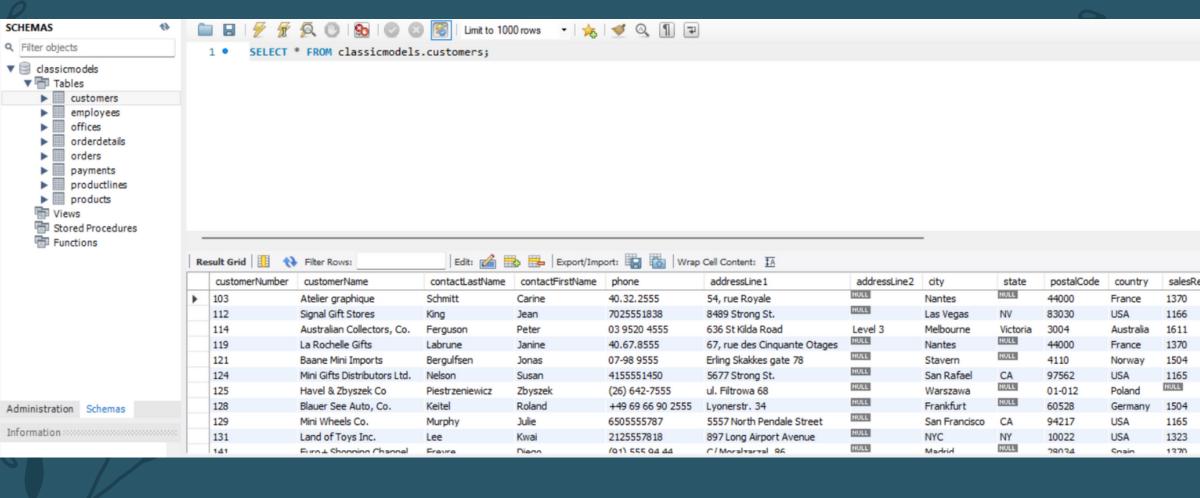
Introduction: Exploring the Classic Model Database

Welcome to the Classic Model Database, a rich and comprehensive dataset designed to capture and analyze data from a classic model business scenario. This relational database provides a structured representation of various aspects of a business, including customers, employees, offices, orders, payments, and products.

Mectine.

Throughout our exploration, we aim to derive meaningful insights by posing diverse queries and analyses on this dataset. From understanding customer credit limits to identifying top-performing products, our exploration will provide valuable perspectives on the business model represented in this classic dataset.

DATABAGE





1. Find the total number of customers in the database.

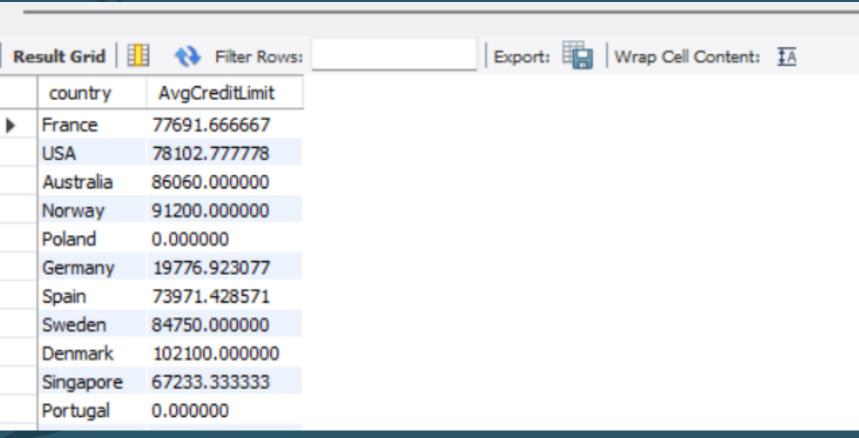
SELECT COUNT(*) AS total_customers FROM classicmodels.customers;





2. Find the average credit limit of customers in each country.

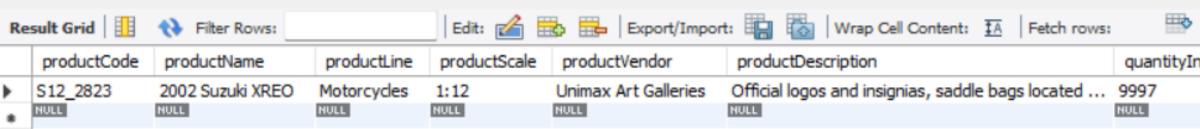
SELECT country, AVG(creditLimit) AS AvgCriditLimit
FROM classicmodels.customers
GROUP BY country;





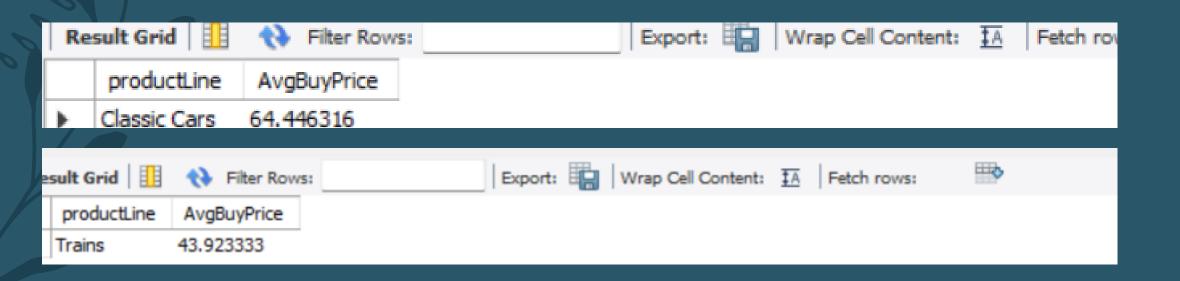
3. Retrieve the details of the products with the highest quantity in stock.

SELECT * FROM classic models.products ORDER BY quantityInStock DESC LIMIT 1;



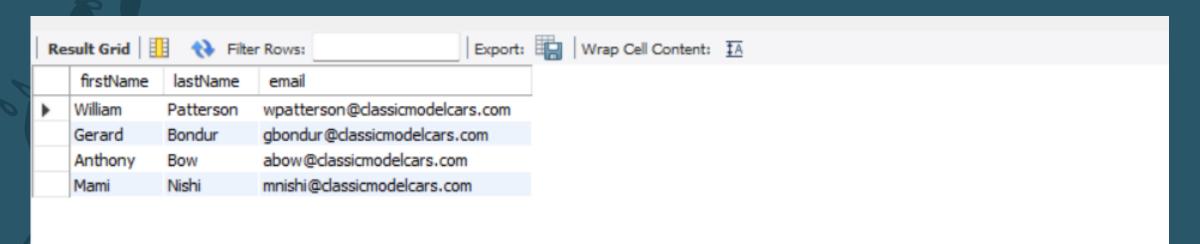


SELECT productLine, AVG(buyPrice) AS AvgBuyPrice
FROM classicmodels.products
GROUP BY productLine
ORDER BY AvgBuyPrice DESC/ASC
LIMIT 1;



5. List the names and email addresses of employees who report to a specific manager (reportsTo).

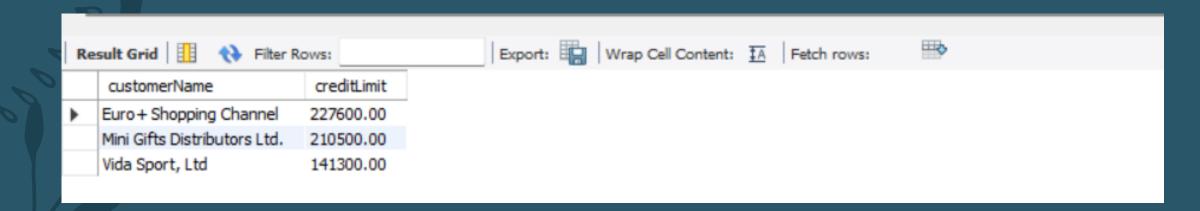
SELECT firstName, lastName, email FROM classicmodels.employees
WHERE reportsTo = 1056;





6. Find the top 3 customers with the highest credit limits.

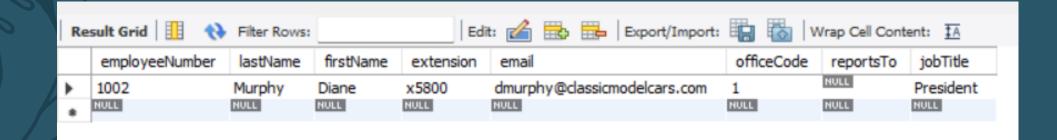
SELECT customerName, creditLimit FROM classicmodels.customers
ORDER BY creditLimit DESC
Limit 3;





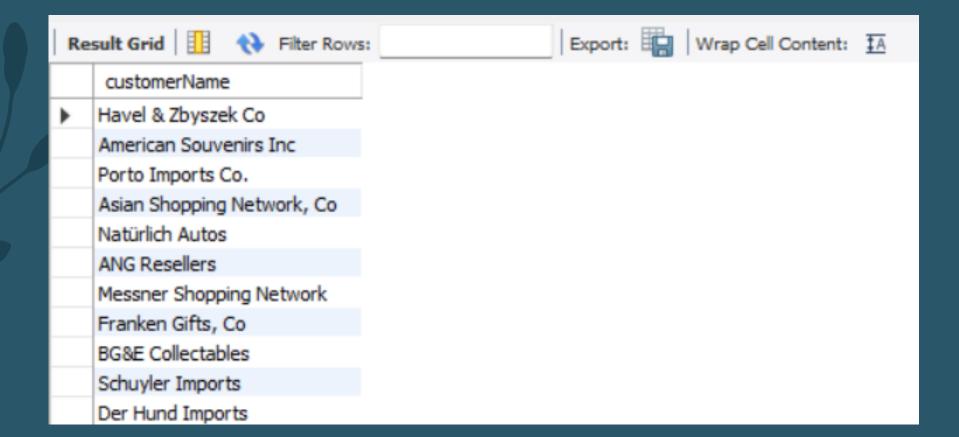
7. List the employees who have not reported to anyone.

FROM classicmodels.employees emp WHERE emp.reportsTo IS NULL;





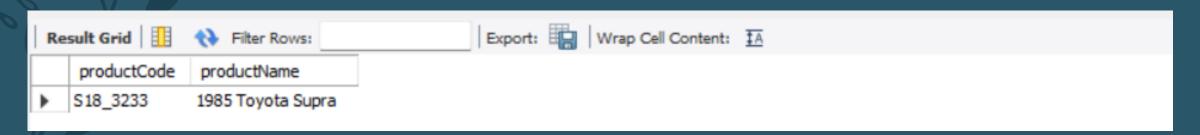
SELECT c.customerName
FROM classicmodels.customers c
LEFT JOIN classicmodels.payments p ON
c.customerNumber = p.customerNumber
WHERE p.customerNumber IS NULL;



9. List the product codes and names of products that have not been ordered.

SELECT p.productCode, p.productName
FROM classicmodels.products p

LEFT JOIN classicmodels.orderdetails od ON
p.productCode = od.productCode
WHERE od.productCode IS NULL;





10. Find the average order quantity for each product line.

SELECT pr.productLine, AVG(od.quantityOrdered) AS AvgOrderQuantity
FROM classicmodels.orderdetails od
JOIN classicmodels.products pr ON od.productCode = pr.productCode
GROUP BY pr.productLine;

