**Lab-3**

1. Write a query that displays all the rows from the Person.Person table where the rows were modified after December 29, 2000. Display the business entity ID number, the name columns, and the modified date.

Solution-

SELECT BusinessEntityID, FirstName, MiddleName, LastName, ModifiedDate FROM Person.Person WHERE ModifiedDate > '2000-12-29';

1. Rewrite the query from question 1 so that it displays the rows that were not modified during December 2000.

Solution-

SELECT BusinessEntityID, FirstName, MiddleName, LastName, ModifiedDate FROM Person.Person WHERE ModifiedDate BETWEEN '2000-12-01' AND '2000-12-31';

3. Write a query that displays the product ID and name for each product from the Production.Product table with the name starting with Chain

Solution-

SELECT ProductID, Name FROM Production.Product WHERE Name LIKE 'Chain%';

1. Write a query that displays the business entity ID number, first name, middle name, and last name from the Person.Person table for only those rows that have E or B stored in the middle name column.

Solution-

SELECT BusinessEntityID, FirstName, MiddleName, LastName FROM Person.Person WHERE MiddleName LIKE '[E,B]';

1. Write a query displaying the order ID, order date, and total due from the Sales.SalesOrderHeader table.

Retrieve only those rows where the order was placed during the month of September 2001 and the total due exceeded $1,000.

Solution- SELECT SalesOrderID, OrderDate, TotalDue FROM Sales.SalesOrderHeader WHERE OrderDate BETWEEN '2001-09-01' AND '2001-09-30' AND TotalDue > 1000;

1. Write a query displaying the sales orders where the total due exceeds $1,000. Retrieve only those rows where the salesperson ID is 279 or the territory ID is 6.

Solution-

SELECT SalesOrderID, OrderDate, TotalDue, SalesPersonID, TerritoryID FROM Sales.SalesOrderHeader WHERE TotalDue > 1000 AND (SalesPersonID = 279 OR TerritoryID = 6);

1. Write a query displaying the ProductID, Name, and Color columns from rows in the Production.Product table. Display only those rows in which the color is not blue.

Solution-

SELECT ProductID, Name, Color FROM Production.Product WHERE Color IS NULL OR Color <> 'Blue';

1. Write a query that returns the business entity ID and name columns from the Person.Person table. Sort the results by LastName, FirstName, and MiddleName.

Solution-

SELECT BusinessEntityID, LastName, FirstName, MiddleName FROM Person.Person

ORDER BY LastName, FirstName, MiddleName;

1. Write a query that displays in the “AddressLine1 (City PostalCode)” format from the Person.Address table.

Solution-

SELECT AddressLine1 + ' (' + City + ' ' + PostalCode + ')' FROM Person.Address;

1. Write a query using the Production.Product table displaying the product ID, color, and name columns. If the color column contains a NULL value, replace the color with No Color.

Solution-

SELECT ProductID, ISNULL(Color,'No Color') AS Color, Name FROM Production.Product;

1. Modify the query written in question 2 so that the description of the product is displayed in the “Name: Color” format. Make sure that all rows display a value even if the Color value is missing.

Solution-

SELECT ProductID, Name + ISNULL(': ' + Color,'') AS Description FROM Production.Product;

1. Write a query using the Sales.SpecialOffer table. Display the difference between the MinQty and MaxQty columns along with the SpecialOfferID and Description columns.

Solution-

SELECT SpecialOfferID, Description, MaxQty - MinQty AS Diff FROM Sales.SpecialOffer;

1. Write a query using the Sales.SpecialOffer table that multiplies the MaxQty column by the DiscountPCT column. If the MaxQty value is null, replace it with the value 10. Include the SpecialOfferID and Description columns in the results.

Solution-

SELECT SpecialOfferID, Description, ISNULL(MaxQty,10) \* DiscountPct AS Discount FROM Sales.SpecialOffer;

1. Write a query that displays the first 10 characters of the AddressLine1 column in the Person.Address table.

Solution-

SELECT LEFT(AddressLine1,10) AS Address10 FROM Person.Address;

1. Write a query that calculates the number of days between the date an order was placed and the date that it was shipped using the Sales.SalesOrderHeader table. Include the SalesOrderID, OrderDate, and ShipDate columns.

Solution-

SELECT SalesOrderID, OrderDate, ShipDate, DATEDIFF(d,OrderDate,ShipDate) AS NumberOfDays FROM Sales.SalesOrderHeader;

1. Write a query that displays only the date, not the time, for the order date and ship date in the Sales.SalesOrderHeader table. (Use any of the styles that return only date)

Solution-

SELECT CONVERT(VARCHAR,OrderDate,1) AS OrderDate, CONVERT(VARCHAR, ShipDate,1) AS ShipDate FROM Sales.SalesOrderHeader;

1. Write a query that adds six months to each order date in the Sales.SalesOrderHeader table. Include the SalesOrderID and OrderDate columns.

Solution-

SELECT SalesOrderID, OrderDate, DATEADD(m,6,OrderDate) Plus6Months FROM Sales.SalesOrderHeader;

1. Write a query that displays the year of each order date and the numeric month of each order date in separate columns in the results. Include the SalesOrderID and OrderDate columns.

Solution-

SELECT SalesOrderID, OrderDate, YEAR(OrderDate) AS OrderYear, MONTH(OrderDate) AS OrderMonth FROM Sales.SalesOrderHeader;

1. Write a statement that generates a random number between 1 and 10 each time it is run.

Solution-

SELECT CAST(RAND() \* 10 AS INT) + 1;

1. Write a query using the Sales.SalesOrderHeader table to display the orders placed during 2001 by using a function. Include the SalesOrderID and OrderDate columns in the results.

Solution-

SELECT SalesOrderID, OrderDate FROM Sales.SalesOrderHeader ORDER BY MONTH(OrderDate), YEAR(OrderDate);

1. Write a query using the Sales.SalesOrderHeader table listing the sales in order of the month the order was placed and then the year the order was placed. Include the SalesOrderID and OrderDate columns in the results.

Solution-

SELECT SalesOrderID, OrderDate FROM Sales.SalesOrderHeader ORDER BY MONTH(OrderDate), YEAR(OrderDate);

1. The HumanResources.Employee table does not contain the employee names. Join that table to the Person.Person table on the BusinessEntityID column. Display the job title, birth date, first name, and last name.

Solution-

SELECT JobTitle, BirthDate, FirstName, LastName FROM HumanResources.Employee AS E INNER JOIN Person.Person AS P ON E.BusinessEntityID = P.BusinessEntityID;

1. The customer names also appear in the Person.Person table. Join the Sales.Customer table to the Person.Person table. The BusinessEntityID column in the Person.Person table matches the PersonID column in the Sales.Customer table. Display the CustomerID, StoreID, and TerritoryID columns along with the name columns

Solution-

SELECT CustomerID, StoreID, TerritoryID, FirstName, MiddleName, LastName FROM Sales.Customer AS C INNER JOIN Person.Person AS P ON C.PersonID = P.BusinessEntityID;

1. Write a query that joins the Sales.SalesOrderHeader table to the Sales. SalesPerson table. Join the BusinessEntityID column from the Sales.SalesPerson table to the SalesPersonID column in the Sales.SalesOrderHeader table. Display the SalesOrderID along with the SalesQuota and Bonus.

Solution-

SELECT SalesOrderID, SalesQuota, Bonus FROM Sales.SalesOrderHeader AS S INNER JOIN Sales.SalesPerson AS SP ON S.SalesPersonID = SP.BusinessEntityID;

1. The catalog description for each product is stored in the Production.ProductModel table. Display the columns that describe the product from the Production.Product table, such as the color and size along with the catalog description for each product.

Solution-

SELECT PM.CatalogDescription, Color, Size FROM Production.Product AS P INNER JOIN Production.ProductModel AS PM ON P.ProductModelID = PM.ProductModelID;

1. Write a query that displays the names of the customers along with the product names that they have purchased. Hint: Five tables will be required to write this query!

Solution-

SELECT FirstName, MiddleName, LastName, Prod.Name FROM Sales.Customer AS C INNER JOIN Person.Person AS P ON C.PersonID = P.BusinessEntityID INNER JOIN Sales.SalesOrderHeader AS SOH ON C.CustomerID = SOH.CustomerID INNER JOIN Sales.SalesOrderDetail AS SOD ON SOH.SalesOrderID = SOD.SalesOrderID INNER JOIN Production.Product AS Prod ON SOD.ProductID = Prod.ProductID;

1. Write a query that displays all the products along with the SalesOrderID even if an order has never been placed for that product. Join to the Sales.SalesOrderDetail table using the ProductID column.

Solution-

SELECT SalesOrderID, P.ProductID, P.Name FROM Production.Product AS P LEFT OUTER JOIN Sales.SalesOrderDetail AS SOD ON P.ProductID = SOD.ProductID;

1. The Sales.SalesOrderHeader table contains foreign keys to the Sales.CurrencyRate and Purchasing.ShipMethod tables. Write a query joining all three tables, making sure it contains all rows from Sales.SalesOrderHeader. Include the CurrencyRateID, AverageRate, SalesOrderID, and ShipBase columns.

Solution-

SELECT CR.CurrencyRateID, CR.AverageRate, SM.ShipBase, SalesOrderID FROM Sales.SalesOrderHeader AS SOH LEFT OUTER JOIN Sales.CurrencyRate AS CR ON SOH.CurrencyRateID = CR.CurrencyRateID LEFT OUTER JOIN Purchasing.ShipMethod AS SM ON SOH.ShipMethodID = SM.ShipMethodID;

1. Write a query to determine the number of customers in the Sales.Customer table.

Solution-

SELECT COUNT(\*) AS CountOfCustomers FROM Sales.Customer;

1. Write a query using the Production.Product table that displays the minimum, maximum, and average ListPrice.

Solution-

SELECT MIN(ListPrice) AS Minimum, MAX(ListPrice) AS Maximum, AVG(ListPrice) AS Average FROM Production.Product;

1. Write a query that shows the total number of items ordered for each product. Use the Sales.SalesOrderDetail table to write the query.

Solution-

SELECT SUM(OrderQty) AS TotalOrdered, ProductID FROM Sales.SalesOrderDetail GROUP BY ProductID;

1. Write a query using the Sales.SalesOrderDetail table that displays a count of the detail lines for each SalesOrderID.

Solution-

SELECT COUNT(\*) AS CountOfOrders, SalesOrderID FROM Sales.SalesOrderDetail GROUP BY SalesOrderID;

1. Write a query using the Production.Product table that lists a count of the products in each product line

Solution-

SELECT COUNT(\*) AS CountOfProducts, ProductLine FROM Production.Product GROUP BY ProductLine;

1. Write a query that displays the count of orders placed by year for each customer using the Sales.SalesOrderHeader table.

Solution-

SELECT CustomerID, COUNT(\*) AS CountOfSales, YEAR(OrderDate) AS OrderYear FROM Sales.SalesOrderHeader GROUP BY CustomerID, YEAR(OrderDate);

1. Write a query that creates a sum of the LineTotal in the Sales.SalesOrderDetail table grouped by the SalesOrderID. Include only those rows where the sum exceeds 1,000.

Solution-

SELECT SUM(LineTotal) AS SumOfLineTotal, SalesOrderID FROM Sales.SalesOrderDetail GROUP BY SalesOrderID HAVING SUM(LineTotal) > 1000;

1. Write a query that groups the products by ProductModelID along with a count. Display the rows that have a count that equals 1.

Solution-

SELECT ProductModelID, COUNT(\*) AS CountOfProducts FROM Production.Product GROUP BY ProductModelID HAVING COUNT(\*) = 1;

1. Write a query using the Sales.SalesOrderHeader, Sales.SalesOrderDetail, and Production.Product tables to display the total sum of products by ProductID and OrderDate.

Solution-

SELECT SUM(OrderQty) SumOfOrderQty, P.ProductID, SOH.OrderDate FROM Sales.SalesOrderHeader AS SOH INNER JOIN Sales.SalesOrderDetail AS SOD ON SOH.SalesOrderID = SOD.SalesOrderDetailID INNER JOIN Production.Product AS P ON SOD.ProductID = P.ProductID GROUP BY P.ProductID, SOH.OrderDate;