Lab 2

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.**
   1. 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 TerritoryIDcolumns along with the name columns.**
   1. SELECT CustomerID, StoreID, TerritoryID, FirstName, MiddleName, LastName

FROM Sales.Customer AS C

INNER JOIN Person.Person AS P ON C.PersonID = P.BusinessEntityID;

1. **Extend the query written in question 2 to include the Sales.SalesOrderHeader table. Display the SalesOrderID column along with the columns already specified. The Sales.SalesOrderHeadertable joins the Sales.Customer table on CustomerID**
   1. SELECT c.CustomerID, StoreID, c.TerritoryID, FirstName, MiddleName,

LastName, SalesOrderID

FROM Sales.Customer AS C

INNER JOIN Person.Person AS P ON C.PersonID = P.BusinessEntityID

INNER JOIN Sales.SalesOrderHeader AS S ON S.CustomerID = C.CustomerID;

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.**
   1. SELECT SalesOrderID, SalesQuota, Bonus

FROM Sales.SalesOrderHeader AS S

INNER JOIN Sales.SalesPerson AS SP

ON S.SalesPersonID = SP.BusinessEntityID;

1. **Add the name columns to the query written in question 4 by joining on the Person.Person table. See whether you can figure out which columns will be used to write the join. You can join the Person.Person table on the SalesOrderHeader table or the Sales.SalesPersontable.**

SELECT SalesOrderID, SalesQuota, Bonus, FirstName, MiddleName, LastName

FROM Sales.SalesOrderHeader AS S

INNER JOIN Sales.SalesPerson AS SP ON S.SalesPersonID = SP.BusinessEntityID

INNER JOIN Person.Person AS P ON SP.BusinessEntityID = P.BusinessEntityID**;**

SELECT SalesOrderID, SalesQuota, Bonus, FirstName, MiddleName, LastName

FROM Sales.SalesOrderHeader AS S

INNER JOIN Sales.SalesPerson AS SP ON S.SalesPersonID = SP.BusinessEntityID

INNER JOIN Person.Person AS P ON S.SalesPersonID = P.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**
   1. 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!**
   1. 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.**
   1. SELECT SalesOrderID, P.ProductID, P.Name FROM Production.Product AS P

LEFT OUTER JOIN Sales.SalesOrderDetail AS SOD ON P.ProductID = SOD.ProductID

1. **Change the previous query so that only products that have not been ordered show up in the query**
   1. SELECT SalesOrderID, P.ProductID, P.Name FROM Production.Product AS P

LEFT OUTER JOIN Sales.SalesOrderDetail AS SOD ON P.ProductID = SOD.ProductID WHERE SalesOrderID IS NULL;

1. **Write a query that returns all the rows from the Sales.SalesPerson table joined to the Sales.SalesOrderHeader table along with the SalesOrderID column even if no orders match. Include the SalesPersonID and SalesYTD columns in the results**
   1. SELECT SalesOrderID, SalesPersonID, SalesYTD FROM Sales.SalesPerson AS SP

LEFT OUTER JOIN Sales.SalesOrderHeader AS SOH ON SP.BusinessEntityID = SOH.SalesPersonID**;**

1. **Change the previous query written so that the salesperson’s name also displays from the Person.Person table.**
   1. SELECT SalesOrderID, SalesPersonID, SalesYTD, FirstName, MiddleName, LastName FROM Sales.SalesPerson AS SP

LEFT OUTER JOIN Sales.SalesOrderHeader AS SOH

ON SP.BusinessEntityID = SOH.SalesPersonID

LEFT OUTER JOIN Person.Person AS P

ON P.BusinessEntityID = SP.BusinessEntityID;

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.**

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 that returns the BusinessEntityID column from the Sales.SalesPerson table along with every ProductID from the Production.Product table.**
   1. SELECT SP.BusinessEntityID, P.ProductID FROM Sales.SalesPerson AS SP CROSS JOIN Production.Product AS P;
2. **Write a query to determine the number of customers in the Sales.Customer table.**
   1. SELECT COUNT(\*) AS CountOfCustomers FROM Sales.Customer;
3. **Write a query that lists the total number of products ordered. Use the OrderQty column of the Sales.SalesOrderDetail table and the SUM function.**
   1. SELECT SUM(OrderQty) AS TotalProductsOrdered FROM Sales.SalesOrderDetail;
4. **Write a query to determine the price of the most expensive product ordered. Use the UnitPricecolumn of the Sales.SalesOrderDetail table.**
   1. SELECT MAX(UnitPrice) AS MostExpensivePrice FROM Sales.SalesOrderDetail;
5. **Write a query to determine the average freight amount in the Sales.SalesOrderHeader table.**
   1. SELECT AVG(Freight) AS AverageFreight FROM Sales.SalesOrderHeader;
6. **Write a query using the Production.Product table that displays the minimum, maximum, and average ListPrice.**
   1. SELECT MIN(ListPrice) AS Minimum, MAX(ListPrice) AS Maximum, AVG(ListPrice) AS Average FROM Production.Product
7. **Write a query that shows the total number of items ordered for each product. Use the Sales.SalesOrderDetail table to write the query.**
   1. 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.**
   1. 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.**
   1. 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.**
   1. SELECT CustomerID, COUNT(\*) AS CountOfSales, YEAR(OrderDate) AS OrderYear

FROM Sales.SalesOrderHeader GROUP BY CustomerID, YEAR(OrderDate);

1. **Write a query that returns a count of detail lines in the Sales.SalesOrderDetail table by SalesOrderID. Include only those sales that have more than three detail lines.**
   1. SELECT COUNT(\*) AS CountOfDetailLines, SalesOrderID FROM Sales.SalesOrderDetail GROUP BY SalesOrderID HAVING COUNT(\*) > 3;
2. **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.**
   1. SELECT SUM(LineTotal) AS SumOfLineTotal, SalesOrderID FROM Sales.SalesOrderDetail GROUP BY SalesOrderID HAVING SUM(LineTotal) > 1000;
3. **Write a query that groups the products by ProductModelID along with a count. Display the rows that have a count that equals 1.**
   1. SELECT ProductModelID, COUNT(\*) AS CountOfProducts FROM Production.Product

GROUP BY ProductModelID HAVING COUNT(\*) = 1;

1. **Change the previous query so that only the products with the color blue or red are included.**
   1. SELECT ProductModelID, COUNT(\*) AS CountOfProducts, Color FROM Production.Product WHERE Color IN ('Blue','Red') GROUP BY ProductModelID, Color HAVING COUNT(\*) = 1;
2. **Write a query using the Sales.SalesOrderDetail table to come up with a count of unique ProductID values that have been ordered.**
   1. SELECT COUNT(DISTINCT ProductID) AS CountOFProductID FROM Sales.SalesOrderDetail;
3. **Write a query using the Sales.SalesOrderHeader table that returns the count of unique TerritoryID values per customer.**
   1. SELECT COUNT(DISTINCT TerritoryID) AS CountOfTerritoryID, CustomerID

FROM Sales.SalesOrderHeader GROUP BY CustomerID

1. **Write a query joining the Person.Person, Sales.Customer, and Sales.SalesOrderHeader tables to return a list of the customer names along with a count of the orders placed.**
   1. SELECT COUNT(\*) AS CountOfOrders, FirstName, MiddleName, LastName

FROM Person.Person AS P

INNER JOIN Sales.Customer AS C ON P.BusinessEntityID = C.PersonID

INNER JOIN Sales.SalesOrderHeader AS SOH ON C.CustomerID = SOH.CustomerID

GROUP BY FirstName, MiddleName, LastName;

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.**
   1. 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;