**Simple SELECT Queries**

1. Write a SELECT statement that lists the customers along with their ID numbers. Include the last

names, first names, and company names.

1. Write a SELECT statement that lists the name, product number, and color of each product.

**Using DISTINCT**

1. Write a SELECT statement that lists all the unique company names.

**Data Filtering**

1. Write a query using a WHERE clause that displays all the employees listed in the

**HumanResources.Employee** table who have the job title Research and Development Engineer.

Display the business entity ID number, the login ID, and the title for each one.

1. Write a query using a WHERE clause that displays all the names in **Person.Person** with the middle name J. Display the first, last, and middle names along with the ID numbers.

**Filtering with Multiple Predicates**

1. Write a query displaying the data from **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.

**Filtering with Wildcards**

1. Write a query like the one in question 1 that displays the products with ‘helmet’ in the name.
2. Change the last query so that the products without helmet in the name are displayed.

**Sorting data in Ascending order**

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.

1. Sort the results by LastName, FirstName, and MiddleName in descending order in the same query.

**Working with Dates**

**Date Format to be used in query : YYYYMMDD**

1. Write a query to display BusinessEntityID, FirstName, MiddleName, LastName, ModifiedDate

in Person.Person where ModifiedDate is BETWEEN 01/12/2005 to 31/12/2006.

1. Write a query to display BusinessEntityID, FirstName, MiddleName, LastName, ModifiedDate

in Person.Person where ModifiedDate is not BETWEEN 01/12/2005 to 31/12/2006.

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.

1. Write a query that displays only the date, not the time, for the order date and ship date in the

**Sales.SalesOrderHeader** table.

1. Write a query that adds six months to each order date in the **Sales.SalesOrderHeader** table.

Include the SalesOrderID and OrderDate columns.

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.

1. Display the month name in same query instead of numeric month.

**Using the IN Operator**

1. Write a query to display SalesOrderID, OrderDate, TotalDue from **Sales.SalesOrderHeader** table WHERE OrderDate IN ('20050901', '20060902', '20070903') AND TotalDue is greater than 1000.

**Using the functions in, where and orderby clause together**

1. Write a query to display SalesOrderID, OrderDate using **Sales.SalesOrderHeader** table WHERE year of Order IN (2005,2006).
2. Write a query to display SalesOrderID, month and year of Order using **Sales.SalesOrderHeader** table with month and year of order in descending order.

**Using CASE function**

1. Use CASE statement to display “Even” when the BusinessEntityID value is an even

number or “Odd” when it is odd. **Hint: Use the modulo operator.**

1. Write a query using the **Sales.SalesOrderDetail** table to display a value (“Under 10” or “10–19” or

“20–29” or “30–39” or “40 and over”) based on the OrderQty value using the CASE function.

Include the SalesOrderID and OrderQty columns in the results.

1. Write a query that displays those rows that have E or B stored in the middle name column.

**Using the Logical Operator**

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.

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 AND the territory ID is 6.

**Working with Nulls**

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.

**SQL Joins**

**Inner joins**

1. Write a query to display JobTitle, BirthDate, FirstName, LastName joining HumanResources.Employee and Person.Person.
2. Write a query to display customerID, StoreID, TerritoryID, FirstName, MiddleName,

LastName, SalesOrderID joining **Sales.SalesOrderHeader**, **Sales.Customer** and **Person.Person** tables.

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!

**Person.Person, Sales.SalesOrderHeader, Sales.SalesOrderDetail, Sales.Customer, Production.Product**

**Left outer joins**

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. Change the query written in previous question so that only products that have not been ordered show up in the query.
2. 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.

**Writing Subqueries**

1. Using a subquery, display the product names and product ID numbers from the

Production.Product table that have been ordered.

1. Change the query written in previous question to display the products that have not been ordered.

**Using UNION function**

1. Write a UNION query that combines the ModifiedDate from **Person.Person** and the HireDate from

**HumanResources.Employee**.

**Using Aggregate Functions**

**COUNT**

1. Determine the number of customers in the **Sales.Customer** table.

**SUM**

1. Write a query that lists the total number of products ordered.

**MIN/MAX/AVG**

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

**Using the GROUP BY Clause**

1. Write a query that shows the total number of items ordered for each product.
2. Write a query using the **Sales.SalesOrderDetail** table that displays a count of the detail lines for each SalesOrderID.
3. Write a query that displays the count of orders placed by year for each customer using the Sales.SalesOrderHeader table.

**Using the HAVING Clause**

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

that have a count that equals 1.

1. Write a query using the Sales.SalesOrderDetail table to come up with a count of unique

ProductID values that have been ordered.

1. Write a query using the Sales.SalesOrderHeader table that returns the count of unique

TerritoryID values per customer.

**Using Operators**

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

**Replacing NULL**

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.

**CAST/CONVERT function**

1. Write a query using the **Production.Product** table displaying a description with the “ProductID:

Name” format. **Hint: You will need to use a function to write this query.**

**ISNULL function**

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.