## Sample queries for Sales Orders database.

1. (Simple SELECT) Find the names and zip codes of all vendors and order them by zip code.

SELECT VendName, VendZipCode
FROM Vendors
ORDER BY VendZipCode;
2. (Simple SELECT with ORDER BY) Find the names and zip codes of all vendors and order them in decreasing order by zip code.

SELECT VendName, VendZipCode
FROM Vendors
ORDER BY VendZipCode DESC;
3. (Simple SELECT with ORDER BY) Find names, phone numbers, and ID’s of all employees, and order them by last name, then first name.

SELECT EmpLastName, EmpFirstName, EmpPhoneNumber, EmployeeID
FROM Employees
ORDER BY EmpLastName, EmpFirstName;
4. (SELECT DISTINCT) Find a list of all states where we have customers.

SELECT DISTINCT CustState
FROM Customers;
5. (SELECT with calculated field) Find names, phone numbers, and cities of all employees, and store the names as EmployeeName. Note that some variants of SQL allow the alias for a field to stand by itself, as in **AS EmployeeName**. However, as I discovered when working through the examples in the Excel Advanced Report Development book, Excel requires the alias to be enclosed in square brackets or single quotes, as in **AS [EmployeeName]** or **AS 'EmployeeName'**. Otherwise, it ignores the aliases completely!

**Access:**
SELECT EmpFirstName & ' ' & EmpLastName AS [EmployeeName], EmpPhoneNumber, EmpCity
FROM Employees;

**SQL Server:**
SELECT EmpFirstName + ' ' + EmpLastName AS [EmployeeName], EmpPhoneNumber, EmpCity
FROM Employees;
6. (SELECT with calculated field) Find the order number and the number of days to ship (stored as DaysToShip) of all orders. Note that the CAST function, which isn't supported in Access, converts a value to a particular data type.

**Access:**
SELECT OrderNumber, ShipDate-OrderDate AS [DaysToShip]
FROM Orders;

**SQL Server:**
SELECT OrderNumber, CAST(ShipDate-OrderDate AS Integer) AS [DaysToShip]
FROM Orders;
7. (WHERE clause with strings) Find the vendor name and phone number of all vendors not in the city of Bellevue.

SELECT VendName, VendPhoneNumber
FROM Vendors
WHERE VendCity<>'Bellevue';
8. (WHERE clause with literal date) Find the order number and ship date of all orders placed on July 19, 1999.

**Access:**
SELECT OrderNumber, ShipDate
FROM Orders
WHERE OrderDate=#7/19/99#;

**SQL Server:**
SELECT OrderNumber, ShipDate
FROM Orders
WHERE OrderDate='7/19/99';
9. (WHERE clause with dates) Find the order number and ship date of all orders where the ship date is before the order date. (There shouldn't be any such orders, but we'll check, just in case.)

SELECT OrderNumber, ShipDate
FROM Orders
WHERE ShipDate<OrderDate;
10. (WHERE clause with numbers) Find the product name and retail price of all products with a retail price less than or equal to $50.

SELECT ProductName, RetailPrice
FROM Products
WHERE RetailPrice<=50;
11. (WHERE clause with LIKE) Find the last name and first name of all customers whose last name starts with P.

**Access:**
SELECT CustLastName, CustFirstName
FROM Customers
WHERE CustLastName LIKE 'P\*';

**SQL Server:**
SELECT CustLastName, CustFirstName
FROM Customers
WHERE CustLastName LIKE 'P%';
12. (WHERE clause with LIKE) Find the vendor name and street address of all vendors with a street address that includes Forest.

**Access:**
SELECT VendName,VendStreetAddress
FROM Vendors
WHERE VendStreetAddress LIKE '\*Forest\*';

**SQL Server:**
SELECT VendName,VendStreetAddress
FROM Vendors
WHERE VendStreetAddress LIKE '%Forest%';
13. (WHERE clase with IN) Find all of the customers who live in California or Oregon.

SELECT CustLastName, CustFirstName
FROM Customers
WHERE CustState IN ('CA','OR');
14. (WHERE clause with BETWEEN and literal dates) Find the order number and order date of all orders with an order date not in the month of August 1999.

**Access:**
SELECT OrderNumber,OrderDate
FROM Orders
WHERE OrderDate NOT BETWEEN #8/1/1999# AND #8/31/1999#;

**SQL Server:**
SELECT OrderNumber,OrderDate
FROM Orders
WHERE OrderDate NOT BETWEEN '8/1/1999' AND '8/31/1999';
15. (WHERE clause with AND) Find the first and last name and city of all customers with a last name that begins with S and live in Seattle. (Parentheses in the WHERE clause aren't required; they are added for clarity only.)

**Access:**
SELECT CustFirstName, CustLastName, CustCity
FROM Customers
WHERE (CustLastName LIKE 'S\*') AND (CustCity='Seattle');

**SQL Server:**
SELECT CustFirstName, CustLastName, CustCity
FROM Customers
WHERE (CustLastName LIKE 'S%') AND (CustCity='Seattle');
16. (WHERE clause with OR) Find the vendor name, phone number, and state of all vendors who are in Washington or California.

SELECT VendName, VendPhoneNumber, VendState
FROM Vendors
WHERE VendState='WA' OR VendState='CA';
17. (INNER JOIN of two tables) Find the category description and product name of all products.

SELECT Categories.CategoryDescription, Products.ProductName
FROM Categories
INNER JOIN Products
  ON Categories.CategoryID = Products.CategoryID;
18. (INNER JOIN using table aliases) This is the same as the previous query, but it uses aliases for the table names. (Any abbreviations would work.) Note that the "AS" keyword is optional and are often omitted. Also, note that once aliases are given, they must be used throughout the query.

SELECT C.CategoryDescription, P.ProductName
FROM Categories AS C
INNER JOIN Products AS P
  ON C.CategoryID = P.CategoryID;
19. (INNER JOIN of multiple tables) Find a list of all customers (first name and last name) who have bought a King Cobra Helmet. Also, list the order date for each such order. (**Note:** The parentheses in the FROM clause are required by Access. They could be omitted for SQL Server. Basically, Access wants the joins to be grouped to indicate that two things are joined at a time.)

SELECT DISTINCT Customers.CustFirstName, Customers.CustLastName, Orders.OrderDate
FROM ((Customers
INNER JOIN Orders
  ON Customers.CustomerID = Orders.CustomerID)
INNER JOIN Order\_Details
  ON Orders.OrderNumber = Order\_Details.OrderNumber)
INNER JOIN Products
  ON Products.ProductNumber = Order\_Details.ProductNumber
WHERE Products.ProductName='King Cobra Helmet';
20. (INNER JOIN of multiple tables using table aliases) This is the same as the previous query, but it uses aliases for the table names. Note that the optional "AS" keyword has now been omitted.

SELECT DISTINCT C.CustFirstName, C.CustLastName, O.OrderDate
FROM ((Customers C
INNER JOIN Orders O
  ON C.CustomerID = O.CustomerID)
INNER JOIN Order\_Details OD
  ON O.OrderNumber = OD.OrderNumber)
INNER JOIN Products P
  ON P.ProductNumber = OD.ProductNumber
WHERE P.ProductName='King Cobra Helmet';
21. (INNER JOIN of multiple tables) Find a list of all customer full names, product names, and order dates for clothing products ordered.

**Access:**
SELECT C.CustFirstName & ' ' & C.CustLastName AS [CustFullName], P.ProductName, O.OrderDate
FROM (((Customers C
INNER JOIN Orders O
  ON C.CustomerID = O.CustomerID)
INNER JOIN Order\_Details OD
  ON O.OrderNumber = OD.OrderNumber)
INNER JOIN Products P
  ON OD.ProductNumber = P.ProductNumber)
INNER JOIN Categories CAT
  ON CAT.CategoryID = P.CategoryID
WHERE CAT.CategoryDescription='Clothing';

**SQL Server:**
SELECT C.CustFirstName + ' ' + C.CustLastName AS [CustFullName], P.ProductName, O.OrderDate
FROM (((Customers C
INNER JOIN Orders O
  ON C.CustomerID = O.CustomerID)
INNER JOIN Order\_Details OD
  ON O.OrderNumber = OD.OrderNumber)
INNER JOIN Products P
  ON OD.ProductNumber = P.ProductNumber)
INNER JOIN Categories CAT
  ON CAT.CategoryID = P.CategoryID
WHERE CAT.CategoryDescription='Clothing';
22. (INNER JOIN of multiple tables, SELECT statement within FROM clause with alias) Find a list of all customers (first name and last name) who have bought Dog Ear Monster Grip Gloves *and* Dog Ear Helmet Mount Mirrors.

SELECT CustGloves.CustFirstName, CustGloves.CustLastName
FROM
    (SELECT DISTINCT C.CustomerID, C.CustFirstName, C.CustLastName
    FROM ((Customers C
    INNER JOIN Orders O
      ON C.CustomerID = O.CustomerID)
    INNER JOIN Order\_Details OD
      ON O.OrderNumber = OD.OrderNumber)
    INNER JOIN Products P
      ON P.ProductNumber = OD.ProductNumber
    WHERE P.ProductName = 'Dog Ear Monster Grip Gloves') AS CustGloves
INNER JOIN
    (SELECT DISTINCT C.CustomerID
    FROM ((Customers C
    INNER JOIN Orders O
      ON C.CustomerID = O.CustomerID)
    INNER JOIN Order\_Details OD
      ON O.OrderNumber = OD.OrderNumber)
    INNER JOIN Products P
      ON P.ProductNumber = OD.ProductNumber
    WHERE P.ProductName = 'Dog Ear Helmet Mount Mirrors') AS CustMirrors
ON CustGloves.CustomerID = CustMirrors.CustomerID;

An alternative way, first creating two named queries, CustGloves and CustMirrors:

SELECT DISTINCT CG.CustLastName, CG.CustFirstName
FROM CustGloves CG
INNER JOIN CustMirrors CM
  ON CG.CustomerID = CM.CustomerID;

For example, the CustGloves query is the following:

SELECT DISTINCT C.CustomerID, C.CustFirstName, C.CustLastName
FROM ((Customers C
INNER JOIN Orders O
  ON C.CustomerID = O.CustomerID)
INNER JOIN Order\_Details OD
  ON O.OrderNumber = OD.OrderNumber)
INNER JOIN Products P
  ON P.ProductNumber = OD.ProductNumber
WHERE P.ProductName = 'Dog Ear Monster Grip Gloves'

1. (COUNT function) Find the number of employees in the state of Washington.

SELECT COUNT(\*) AS [TotalWashEmployees]
FROM Employees
WHERE EmpState = 'WA';
2. (COUNT function) Find the number of customers who have listed their phone number.

SELECT COUNT(CustPhoneNumber) AS [NumWithKnownPhone]
FROM Customers;
3. (COUNT function with DISTINCT option) Find the number of states where we have customers. **Note**: This and the next example illustrate a case where SQL Server implements the ANSI standard, whereas Access doesn't. Therefore, a workaround is required with Access. This uses a subquery (last two lines). Subqueries are seen in a few of the remaining examples.

**Access:**
SELECT COUNT(CustState) AS [NumUniqueStates]
FROM
  (SELECT DISTINCT CustState
  FROM Customers);

**SQL Server:**
SELECT COUNT(DISTINCT CustState) AS [NumUniqueStates]
FROM Customers;
4. (COUNT function with DISTINCT option) Find the number of cities in Oregon where we have customers.

**Access:**
SELECT COUNT(CustCity) AS [NumUniqueOregonCities]
FROM
  (SELECT DISTINCT CustCity
  FROM Customers
  WHERE CustState = 'OR');

**SQL Server:**
SELECT COUNT(DISTINCT CustCity) as NumUniqueOregonCities
FROM Customers
WHERE CustState='OR';
5. (SUM function) Find the total inventory retail value of all of our products.

SELECT SUM(RetailPrice\*QuantityOnHand) AS [TotalInventoryValue]
FROM Products;
6. (AVG function) What is the average retail price of a mountain bike?

SELECT AVG(RetailPrice) AS [AveragePrice]
FROM Products
WHERE ProductName LIKE '%Mountain Bike%';
7. (AVG function) Find the average item total for order 64.

SELECT AVG(QuotedPrice\*QuantityOrdered) AS [AvgItemTotal]
FROM Order\_Details
WHERE OrderNumber=64;
8. (MAX function) What is the largest line item total for order 64?

SELECT MAX(QuotedPrice\*QuantityOrdered) AS [LargestItemTotal]
FROM Order\_Details
WHERE OrderNumber=64;
9. (MAX function with dates) What was the date of our most recent order?

SELECT Max(OrderDate) AS [MostRecentOrderDate]
FROM Orders;
10. (Multiple summary functions) How many different products were ordered on order number 553 and what was the total cost of that order?

SELECT COUNT(ProductNumber) AS [TotalProductsPurchased], SUM(QuotedPrice\*QuantityOrdered) AS [OrderAmount]
FROM Order\_Details
WHERE OrderNumber=553;
11. (Subquery in WHERE clause) List the product names and numbers that have a quoted price greater than or equal to the overall average retail price in the products table.

SELECT DISTINCT P.ProductName
FROM Products P
INNER JOIN Order\_Details OD
  ON P.ProductNumber = OD.ProductNumber
WHERE OD.QuotedPrice>=
  (SELECT AVG(RetailPrice)
  FROM Products);
12. (GROUP BY statement) List for each customer and order date the customer name and the total cost of items ordered on each date.

SELECT C.CustFirstName, C.CustLastName, O.OrderDate, SUM(OD.QuotedPrice\*OD.QuantityOrdered) AS [TotalCost]
FROM (Customers C
INNER JOIN Orders O
  ON C.CustomerID = O.CustomerID)
INNER JOIN Order\_Details OD
  ON O.OrderNumber = OD.OrderNumber
GROUP BY C.CustFirstName, C.CustLastName, O.OrderDate;
13. (GROUP BY statement) Display for each product the product name and the total sales.

SELECT P.ProductName, Sum(OD.QuotedPrice\*OD.QuantityOrdered) AS [TotalSales]
FROM Products P
INNER JOIN Order\_Details OD
  ON P.ProductNumber = OD.ProductNumber
GROUP BY P.ProductName;
14. (GROUP BY statement) List for each vendor the number of products sold.

SELECT V.VendName, Count(PV.ProductNumber) AS [CountOfProductNumber]
FROM Vendors V
INNER JOIN Product\_Vendors PV
  ON V.VendorID = PV.VendorID
GROUP BY V.VendName;
15. (GROUP BY statement) List each vendor and the average (by vendor) of the number of days to deliver products.

SELECT V.VendName, Avg(PV.DaysToDeliver) AS [AvgDelivery]
FROM Vendors V
INNER JOIN Product\_Vendors PV
  ON V.VendorID = PV.VendorID
GROUP BY V.VendName;
16. (GROUP BY and HAVING statements) List for each customer and order date the customer name and the total cost of items ordered, but only if this total cost is greater than $1,000.

SELECT C.CustFirstName, C.CustLastName, O.OrderDate, SUM(OD.QuotedPrice\*OD.QuantityOrdered) AS [TotalCost]
FROM (Customers C
INNER JOIN Orders O
  ON C.CustomerID = O.CustomerID)
INNER JOIN Order\_Details OD
  ON O.OrderNumber = OD.OrderNumber
GROUP BY C.CustFirstName, C.CustLastName, O.OrderDate
HAVING SUM(OD.QuotedPrice\*OD.QuantityOrdered) > 1000;
17. (GROUP BY and HAVING statements with subquery in HAVING clause) List each vendor and the average (by vendor) of the number of days to deliver products, but only list those where this number is greater than the average delivery days of all vendors.

SELECT V.VendName, Avg(PV.DaysToDeliver) AS [AvgDelivery]
FROM Vendors V
INNER JOIN Product\_Vendors PV
  ON V.VendorID = PV.VendorID
GROUP BY V.VendName
HAVING Avg(PV.DaysToDeliver)>
  (SELECT Avg(DaysToDeliver)
  FROM Product\_Vendors);
18. (CASE statement and some useful functions. Note: I googled for Data functions in SQL and found the following interesting site: <http://www.informit.com/articles/article.asp?p=31453&rl=1>.) For all items in each order, list the month, day, and year of the order date, whether each order was shipped late (assuming late means that the ship date was at least 3 days after the order date), the product name, the invoice amount, and whether the retail price is different from the quoted price on the order. (As I found when I tried this in Access, it doesn't work. Access evidently doesn't allow the CASE construct, whereas SQL Server does.)

SELECT Day(O.OrderDate) AS [Order Day], Month(O.OrderDate) AS [Order Month], Year(O.OrderDate) AS [Order Year],
  CASE WHEN O.ShipDate - O.OrderDate >= 3 THEN 'Late' ELSE 'On Time' END AS [Shipping Delay],
  P.ProductName, OD.QuotedPrice \* OD.QuantityOrdered AS [Invoice Amount],
  CASE WHEN OD.QuotedPrice > P.RetailPrice THEN 'Increased' WHEN OD.QuotedPrice < R.RetailPrice THEN 'Decreased' ELSE 'Same' END AS [Price Change]
FROM (Orders O INNER JOIN Order\_Details OD ON O.OrderNumber = OD.OrderNumber)
  INNER JOIN Products P ON OD.ProductNumber = P.ProductNumber
ORDER BY O.OrderDate, O.ShipDate;