NWeHealth, The University of Manchester

Essentials of SQL

Exercise Booklet WITH ANSWERS
Queries modified from various internet resources including SQLZoo.net

Dr. Georgina Moulton
Education and Development Fellow
11/06/2010

SQL Exercises

Remember at each stage to comment your query and save it.

Exercise 1
You should have the AdventureWorksLT database schema. From the information provided can you identify what is a primary key, a foreign key and the type of relationship between the tables.

Exercise 2
Scenario:
AdventureWorks management is attempting to learn more about their customers and they would like to contact each one by email and/or phone to see whether they would complete a survey. You have been asked to produce a list of customers.

The main tasks in this exercise are:

Identify which table contains the information you require
Check what fields the table holds
Generate a table by using the SELECT statement with the FROM clause.

Answer

```
USE AdventureWorksLT;

SELECT SalesLT.Customer.FirstName, SalesLT.Customer.LastName,
SalesLT.Customer.EmailAddress,SalesLT.Customer.Phone
FROM SalesLT.Customer;
```

Exercise 3
Scenario
The marketing department needs a list of the top 15 most expensive items ordered by product number, which includes the fields: product number, list price and product name.

Task 1: Use the ORDER BY clause to format a result set
Identify the correct table in the database that contains the information you need.
Write a SELECT statement that includes, product name, list price and product number.
Sort the row by the list price in descending order column.
Execute the query and browse the result set.

Task 2: Use the ORDER BY and the DESC keyword to sort the list price
Add to the SELECT statement in Task 1, sort by product number in ascending order.
Look at the new result set.

Task 3: Add the TOP statement to get the most expensive items
Add the TOP statement to get the 15 most expensive items.

What is the most expensive item? What is the 15th most expensive item?

Answer

Exercise 4
Scenario
The marketing department (again!) want a distinct list of customers for their campaign. In their list they need the their full name and the company they are affiliated to.

Task 1: Identify the table

Task 2: Write the SELECT statement that retrieves the correct fields for marketing

Task 3: Make sure the list is a distinct list of customers

Answer:
```
Show the first name
--and the email address of customer with CompanyName 'Bike World'

SELECT DISTINCT FirstName, LastName, CompanyName FROM SalesLT.Customer;
```

Exercise 5a
Scenerio
We wish to contact the company 'Bike World' via email. Get the name and email address for this company. Retrieve this details from the correct table using a SELECT statement and the WHERE clause.

Answer
```
Show the first name
--and the email address of customer with CompanyName 'Bike World'

SELECT FirstName, LastName, EmailAddress FROM SalesLT.Customer
WHERE CompanyName='Bike World';
```

Exercise 5b
Scenario
Select all customer names that have SalesPersons jillian or jose.

Answer:
```
SELECT FirstName, LastName, SalesPerson FROM SalesLT.Customer
WHERE SalesPerson LIKE '%jil%' OR SalesPerson LIKE '%jos%';
```

Exercise 6
Scenario
The marketing department is conducting an audit of catalog inventory of socks and tights. In order to increase sales for these products, they have determined that most a willing to spend between £7.00 and £100.00 for these items. They also think that the most popular sizes bought are M and L. They have requested a list of all items the company carries within these parameters.

Task 1: Using a Comparison operator

Identify the table that holds the information you want to retrieve.

Generate a table using the SELECT statement using a comparison operator so that the ListPrice must be lower than £100.

Look at your results. *How many data entries do you retrieve?*

Task 2: Using the AND and LIKE operators

Building on the query in task 1, add that the Product Name column contains the string 'Sock'.
Look at the results. *How many data entries do you retrieve?*

Task 3: Using the OR operator

Change the query from Task 2 so that you include rows with the Tights as well as Socks in the Name Column.
Execute the query and look at the results. Browse the result set and note the additional rows for tights.

Task 4: Using more comparison operators!

Change the query to select rows to display all socks that are now between list prices £7.00 and £100.00.

Browse the results and note the number of rows that have changed and the column data that matches the new search conditions.

Task 4: Adding a condition that limits 'tights' to M and L

Add a search condition that limits the tights to sizes 'M' and 'L'.
Execute the query and see how the results have changed. How have they changed?

Answer

Exercise 7
Scenario
The Finance department are investigating again, especially the postal charges that are directly related to weight and size of the product. They would like a list of all products so they can make corrections in the system.

The main task for this exercise are as follows:

- Generate a table using the SELECT statement with the NULL function
- Generate a table using the SELECT statement with the IS NULL function
- Generate a table using the SELECT statement with the ISNULL function to rename values
- Use the CASE statement to rename values for different items
- Generate a table using the SELECT statement with the ISNULL function and the COALESCE and CONVERT functions

Task 1: Using the NULL function
Write a query that retrieves rows from the product table that includes, product number, name, size and weight. Use the = operator to check the Size and Weight for a NULL value.
Browse the result set and note the total number of rows. How many are there?

Task 2: Using the IS NULL function
Change the statement to IS NULL to select rows with a value of NULL in either the Size or Weight columns.

Task 3: Using the ISNULL function to rename values
In a new query window, enter and execute a SELECT statement that:
Access the product table
Displays the Product Number, Name, Size and Weight columns
Use the ISNULL() function to display 'NA' when a NULL value is encountered in the Size column
Browse the result and note the additional column and values
Also make a note of the column headings

Task 4 Using the CASE statement to substitute size and weight values for products

If the size value is null then substitute the value with NA.
Using the CASE statement, substitute the weights with specific values. If the weight is equal to null then introduce the following weights for these items:

Helmet	4.00
Fork	10.00
Socks	1.00
Jersey	2.00
Any other item	2.00

Task 5: Generate a table using the SELECT statement with the ISNULL function and the COALESCE and CONVERT functions

Rewrite the statement from task 3 using the COALESCE function to create a new column named Measurement so that:
If the Weight column has a value it is shown in the Measurement column
If the Weight column is NULL, but the Size column is not NULL, display the value in the Measurement column
If both columns have NULL values display 'NA'

Exercise 8
Scenario
The products team are updating the system and need to make sure the sell start date, and discontinued date are in the system for all products that have a defined sell end date. If there is no discontinued date then they are going to insert the date that is 10 years ahead of today's date. In addition, they also want to know the number of days that have elapsed since the sell end date and today. They have passed the job over to you!
Create a table with headers that matches the requirements defined by the products team.

Answer:

```
SELECT Name, SellStartDate, SellEndDate, CAST (GETDATE()-SellEndDate AS
INT) AS TimePassed,
ISNULL(DiscontinuedDate, DATEADD(year,10,GETDATE()))
FROM SalesLT.Product
WHERE SellEndDate IS NOT NULL;
```

Exercise 9

Scenario

The Finance department want to know what is the total number of sales (in pounds) they have made each year. Write the SELECT Statement:

Task 1: Identify table for information required

Task 2: Identify a function that will allow you to retrieve the Year from the OrderDate

Task 3: Use the SUM function to add the TotalDue

Task 4: Order by and group by Year

Answer

```
SELECT YEAR=YEAR(OrderDate),TotalSales=SUM(TotalDue)
FROM SalesLT.SalesOrderHeader
GROUP BY YEAR(OrderDate) -- grouping on computed field
ORDER BY YEAR;
```

Exercise 10

Scenario

You have been asked by the sales department to provide the average price for each product category.

Answer

```
SELECT Name, ListPrice, StandardCost,
((ListPrice-StandardCost)/StandardCost)*100 as PercentageCostDiff
FROM
SalesLT.Product;
```

Exercise 11

Scenario

Give the CompanyName of those customers with orders over £100000. Include the subtotal plus tax plus freight. This query requires a join statement.

Task 1: Identify tables you need to get required information
Task 2: Identify common fields that link the tables together
Task 3: Write SELECT statement that mirrors the tasks above with the joins being in the FROM statement

Task 4: Add WHERE statement to make sure orders are over £100000.
Task 5: Group results by Company Name.

Answer

```
--Give the CompanyName of those customers with orders over $100000.
--Include the subtotal plus tax plus freight.
```

```
select * from SalesLT.SalesOrderHeader
WHERE SalesLT.SalesOrderHeader.TotalDue > 100000;

SELECT c.CompanyName
FROM SalesLT.Customer AS c
INNER JOIN SalesLT.SalesOrderHeader AS soh
ON (c.CustomerID=soh.CustomerID)
INNER JOIN SalesLT.SalesOrderDetail as sod
ON (sod.SalesOrderID=soh.SalesOrderID)
WHERE soh.TotalDue > 100000
GROUP BY  C.CompanyName;
```

Exercise 12

Scenario

Where did the racing socks go? List the product name and the CompanyName for
-- all Customers who ordered ProductModel 'Racing Socks'.

Task 1: Identify tables you need to get required information
Task 2: Work out the common fields that link the tables together
Task 3: Write the SELECT statement that joins the tables together using the common fields
Task 4: Add a WHERE statement to restrict the query to Racing Socks.

Answer:

```
--Find the number of left racing socks ('Racing Socks, L')
--ordered by CompanyName 'Riding Cycles'

--SELECT * from SalesLT.SalesOrderDetail;

SELECT p.Name, sod.OrderQty, c.CompanyName, sod.SalesOrderID,
p.ProductID
FROM SalesLT.Customer as c
INNER JOIN SalesLT.SalesOrderHeader as soh
ON(c.CustomerID=soh.CustomerID)
INNER JOIN SalesLT.SalesOrderDetail as sod
ON (sod.SalesOrderID=soh.SalesOrderID)
INNER JOIN SalesLT.Product as p
ON (p.ProductID = sod.ProductID)
WHERE c.CompanyName = 'Riding Cycles'
AND p.Name='Racing Socks, L';
```

Exercise 13

Scenario

Can you show the CompanyName for all customers with an address in City 'Dallas'?

Answer:

```
--Show the CompanyName for all customers with an address in City
'Dallas'.

 SELECT c.CompanyName, a.City
 From SalesLT.Address AS a
 INNER JOIN SalesLT.CustomerAddress AS ca
 ON (a.AddressID=ca.AddressID)
```

```
 INNER JOIN SalesLT.Customer as c
 ON(c.CustomerID=a.AddressID)
 WHERE a.City='Dallas';
```

Exercise 14

Scenario

We have been asked by the Products team for some information about every product and its description for those that have a culture originating in English (En). This information is required so that they know what is coming from England. They are not really interested at this stage about other cultures, but they could be in future requests.

Task 1: Identification of tables for information

As usual, identify the tables that are required for the product information.

Task 2: Identify common fields in tables to link them together

Identify the common fields in the tables identified in task 1 that link them together. Write the query that joins them together and selects the product identifier, culture, product description, product model name

Task 3 :Filtering for the 'en' culture

Adding to the query in task 2, retrieve only products that come from an English (en) culture.

Answer

```
SELECT p.ProductID, p.Name, pm.Name AS ProductModel,
pmx.Culture, pd.Description
FROM   SalesLT.Product AS p
INNER JOIN SalesLT.ProductModel AS pm
ON p.ProductModelID = pm.ProductModelID
INNER JOIN SalesLT.ProductModelProductDescription AS pmx
ON pm.ProductModelID = pmx.ProductModelID
INNER JOIN SalesLT.ProductDescription AS pd ON
pmx.ProductDescriptionID = pd.ProductDescriptionID
WHERE pmx.Culture = 'en';
```

Exercise 15

Scenario

We have been asked by the AdventureWorks Finance Department to find out the total amount due from two customers Walter Brian and Walter Mays on all orders they have placed. It is worth noting each time a customer places an order they are treated as a new customer, so they get a separate record.

Task 1: Investigating the Customer table for customers who have first name Walter

Using a simple SELECT statement retrieve all customers who have first name Walter. What do you notice? How many people have the first name?

Task 2: Grouping the customer records

Change the query from task 1 so that you group the records according to their last name.

Task 3: Identify table that you need to get the TotalDue information and join tables

Having identified the table you need to get the TotalDue information from, identify the common field that links that and the Customer table together.

Using the query from task 2, add the join statement that joins the two tables together

Task 4: Adding the TotalDue information using the SUM function

For each customer sales order, there is a total due, thus we need to add the total due figures together to produce one figure for the finance department.

Introduce in the SELECT statement a clause that adds the total due figures together and displays this figure.

Answer

```
SELECT C.LastName, Sum(SOH.TotalDue) as totaldue
FROM SalesLT.Customer AS C
JOIN SalesLT.SalesOrderHeader AS SOH
ON C.CustomerID = SOH.CustomerID
WHERE FirstName = 'Walter'
GROUP BY C.LastName, TotalDue;
```

Exercise 16

Scenario

The AdventureWorks team have been rung up by a customer ‘Mary’ and have jotted down a message that she would like a catalogue. Unfortunately, the customer’s last name was not recorded, thus they do not know where to send the catalogue! We have been asked to produce a list of all customers and their addresses with first name Mary. They need the information that will allow them to send a catalogue; for example, title, full name, company name and full address, including postal code etc. in alphabetical order for the last name.

Task 1:Investigating the tables to use

Identify the tables that contain the customer and address information for the mailing labels.
Conduct a query that will allow you to see all contents of each of the tables.

Task 2: Identifying customers that have first name ‘Mary’ and order according to last name

Using the Customer table, identify all customers that have first name ‘Mary’
Restrict the query to the information you require for mailing labels and make sure they are alphabetically ordered by last name.

Task 3: Retrieving the customer addresses

Look at how the address and customer tables can be linked together (hint: it may require the link of another table – remember you are looking for common fields).

Identify the fields that you require for the mailing list and note which table they come from.
Using the query from task 2, change it so, it can include the address fields.
Hint: this will require joining tables together!
Use aliases when possible to refer to the tables as it makes the query easier to read.

Task 4: Format the result by concatenating strings and assign column new names
Using the query above, concatenate the name together for one column called 'Customer Name'and a column for each of the following information:
Address line 1
Address line 2
City, State Province, Postal Code as AddressLine3
CountryRegion

Task 5: Alter the query and use the SUBSTRING function
Instead of displaying the first name in full as Mary, we now just want to display the first name as initial. Using the query form the previous task, modify it by using the substring function, to just pick out the initial M for Mary.

Answer

```
USE AdventureWorksLT;

SELECT C.FirstName, C.MiddleName, C.LastName, C.CompanyName,
A.AddressLine1, A.AddressLine2, A.City, A.StateProvince,
A.CountryRegion, A.PostalCode
FROM SalesLT.Customer AS C
JOIN SalesLT.CustomerAddress AS CA ON C.CustomerID = CA.CustomerID
JOIN SalesLT.Address AS A ON CA.AddressID = A.AddressID
WHERE C.FirstName = 'Mary'
ORDER BY LastName;
```

Exercise 17
Scenario
AdventureWorks have asked you to find all products with a minimum standard cost (that is equal to the product standard cost) for products in colours Blue, Yellow and Black.
This exercise is about writing sub-queries, so we will write the query in two sections.

Part 1: Inner query
Write a SELECT statement that retrieves the minimum standard cost for products grouped in colour as Blue, Yellow and Black. Make sure the standard cost is above 0.0.

Part 2: Outer query
Write a SELECT statement that retrieves the product name, list price, colour from the product table and minimum standard cost (which will come from the inner query of part 1).
Now add an inner join statement that has the part 1 query as the table as the first part of the expression. Make sure this is in brackets. Give this statement and alias, such as msc.
For the ON part of the join statement, make sure the colour from the inner query is the same as the colour from the outer query AND the minimum standard cost of the msc table, is the same as the standard cost of the product.

Exercise 18

Scenario

For each product category in each colour, the products department want to know how many products come in each colour and the average list price. Again this is a subquery exercise that includes both joins, group by and aggregate functions.

Part 1: Inner query 1 – (Derived table col)

Write a SELECT statement that retrieves the product category identifier, colour count and average list price for each product category identifier and colour from the SalesLT.Product table. Also in your statement, make sure that if the colour is not recorded than substitute with 'N/A'.

Part 2: Outer query

In this outer query we are to join the product category table onto the derived table col using an inner join on product category identifiers.

Write a SELECT statement that retrieves the product category name, colour, colour count and average list price. In the FROM statement insert inner query 1 in brackets and make sure you give it the alias as col.

Now do an inner join linking to the product category table, on the fields product category identifier.

Order the results by product category name and colour.

Answer

```
SELECT Category = Name, Color,ColorCount,AvgListPrice
FROM (SELECT   ProductCategoryID, -- grouping column
Color = COALESCE(Color,'N/A'), -- grouping column with transformation
ColorCount = COUNT(* ),        -- aggregate function
AvgListPrice = AVG(ListPrice)  -- aggregate function
FROM     SalesLT.Product
GROUP BY ProductCategoryID, Color) x -- derived table (subquery)
INNER JOIN SalesLT.ProductCategory psc
ON psc.ProductCategoryID = x.ProductCategoryID
ORDER BY Category,
Color;
```

Hard Exercise 19

Scenario

How many products in ProductCategory 'Cranksets' have been sold to an address in 'London'?

Answer:

```
select distinct COUNT(p.ProductID) as "Num Products"
from SalesLT.Product p
inner join SalesLT.ProductCategory pc on pc.ProductCategoryID =
p.ProductCategoryID
where pc.Name = 'Cranksets'
and p.ProductID in (
      select prod.ProductID
      from SalesLT.Product prod
      inner join SalesLT.SalesOrderDetail sod on sod.ProductID =
prod.ProductID
```

```
      inner join SalesLT.SalesOrderHeader soh on soh.SalesOrderID =
sod.SalesOrderID
      inner join SalesLT.[Address] ship on ship.AddressID =
soh.ShipToAddressID
      -- question is a bit vague, should it be the customer address,
the shipping address or billing address?
      where ship.City = 'London'
)
```

Hard Exercise 20

Use the SubTotal value in SaleOrderHeader to list orders from the largest to the smallest. For each order show the CompanyName and the SubTotal and the total weight of the order.

Answer

```
Use AdventureWorksLT;

select cust.CompanyName, soh.SubTotal, SUM(p.Weight) as "Total Weight"
from SalesLT.SalesOrderHeader soh
inner join SalesLT.Customer cust on cust.CustomerID = soh.CustomerID
inner join SalesLT.SalesOrderDetail sod on sod.SalesOrderID =
soh.SalesOrderID
inner join SalesLT.Product p on p.ProductID = sod.ProductID
group by cust.CompanyName, soh.SubTotal
order by soh.SubTotal desc;
```

Optional Exercises

Can you answer the following question?
How many items with ListPrice more than $1000 have been sold?

Answer

```
--How many items with ListPrice more than $1000 have been sold?
USE AdventureWorksLT;

SELECT COUNT (*) AS NumberOfItems
FROM SalesLT.SalesOrderHeader as soh
INNER JOIN SalesLT.SalesOrderDetail AS sod
ON (sod.SalesOrderID=soh.SalesOrderID)
INNER JOIN SalesLT.Product as p
ON(p.ProductID=sod.ProductID)
WHERE p.ListPrice > 1000;

SELECT p.Name,p.ListPrice
FROM SalesLT.SalesOrderHeader as soh
INNER JOIN SalesLT.SalesOrderDetail AS sod
ON (sod.SalesOrderID=soh.SalesOrderID)
INNER JOIN SalesLT.Product as p
ON(p.ProductID=sod.ProductID)
WHERE p.ListPrice > 1000;
```

A "Single Item Order" is a customer order where only one item is ordered. Show the SalesOrderID and the UnitPrice for every Single Item Order.

Answer

```
select sod.SalesOrderID, sod.UnitPrice
```

```
from SalesLT.SalesOrderDetail sod
inner join SalesLT.SalesOrderHeader soh on soh.SalesOrderID =
sod.SalesOrderID
where sod.SalesOrderID in (
      select s.SalesOrderID as ordernums
      from SalesLT.SalesOrderDetail s
      group by s.SalesOrderID
      having COUNT(*) = 1
      )
```