Chapter 05 In-class Lab Assignment

ISTA-420, T-SQL Fundamentals

In-class Lab — Table expressions

Use table valued subqueries to execute the following queries. Note that it may be possible to use joins and sets to do the same thing.

Using the TSQLV4 database, part 1

1. Write a report returning the order id, the shipper name, the order date and the city shipped to using a derived table
2. I need a report showing how many orders each customer made during each year. Return the customer id, the year, and the number of orders the customer placed during that year. Use a derived table.
3. What is the growth or decline in the number of orders taken year by year? Your report should contain the year, this years order's, last year's orders, and the difference between the two. Join two derived tables.
4. Using a CTE, give e a list of all our non-American suppliers. I need the supplier id, the supplier name, and the country. Alphabetize the countries.
5. What is the growth or decline in the number of orders taken year by year? Your report should contain the year, this years order's, last year's orders, and the difference between the two. Join two CTEs.
6. Create a view that aggregates the number of orders per customer per year. Then run a query against the view sorting the customers by customer id. Don't forget to drop the view if one exists before you create it. Use the DBO schema.
7. Create a view similar to the Production.Products table, but include the category name as a column. Use the DBO schema. This is an example of denormalizing a database table. After going to all the trouble of normalizing a database, can you think of a good reason to denormalize it?
8. Create an inline table valued function that takes a product id as an argument and returns the sales of the product by month. Disregard the year. We want to see what products are more commonly sold durig particular months. Use the DBO schema. Run a query against your TFV to see if it works.

Using the Northwind database

1. List the number of orders by each customer who lives in the United States using a CTE. Sort from highest to lowest.
2. List the product name and the number of each product from a German supplier sold to a customer in Germany using a CTE. Sort from highest to lowest.
3. Prepare an employee report showing the name of each employee, the number of employees they supervise, and the name of their supervisor using a CTE. Sort by the number of employees supervised.

4. One purpose of views is to *denormalize* databases for the purpose of efficiency, both machine efficiency and programmer efficiency. Creating denormalized objects can turn complex queries into simple ones. For example, suppose you needed a list of all employees who took orders for a specific customer, or all customers who were served by a specific employee. You can create a "table" as a view that contains distinct pairs of customers and employees. This is somewhat complex, so do this in steps.

   (a) Create a query that returns every distinct customer/employee pair.

   (b) Use that query to write another query turning the customerid, customername, and customercontact, and the employeeid, firstname, and lastname.

   (c) Make sure you drop any view that might exist.

   (d) Create a view based on your query.

   (e) Write a report listing all customers served by employee 7, Robert King.

   (f) Write a report listing all employees who served customer CHOPS, Chop-suey Chinese.

   (g) Drop the view.

Using the TSQLV4 database

Use the book's database, TSQLV4, and do the exercises 1 through 6, beginning on page 183. The solutions are in the book beginning on page 188.

Solutions to the lab queries

Attempt to write the queries before you look at the solutions. Do not look at the solutions before you attempt to write the query.

TSQLV4 queries, part 1

```
1  select so.orderid, so.companyname as shipper, so.orderdate, so.shipcity
2   from
3    (select s.shipperid, s.companyname, o.orderdate, o.shipcity, o.orderid
4      from Sales.Shippers s join Sales.Orders o
5      on s.shipperid = o.shipperid) so;
6
7  select yo.custid, yo.order_year, count(yo.order_year) as year_total
8    from (
9      select o.custid, year(o.orderdate) as order_year from Sales.Orders o) as yo
10   group by yo.custid, yo.order_year
11   order by yo.custid;
12
13 select cy.order_year, cy.new_orders, ly.old_orders, cy.new_orders - ly.old_orders as
       increase
14   from
15     (select year(orderdate) as order_year, count(orderid) as new_orders
16         from Sales.Orders group by year(orderdate)) as cy
17   left outer join
18         (select year(orderdate) as order_year, count(orderid) as old_orders
19         from Sales.Orders group by year(orderdate)) as ly
20   on cy.order_year = ly.order_year + 1;
21
22 with for_supps as (
23   select supplierid, companyname, country
24     from Production.Suppliers
25         where country not like 'USA')
26   select * from for_supps
27   order by country;
28
29
30 with thisyear as (
31   select year(orderdate) as curyear, count(orderid) as curorder
32     from Sales.Orders group by year(orderdate)),
33   lastyear as
34   (
35   select year(orderdate) as lastyear, count(orderid) as lastorder
36     from Sales.Orders group by year(orderdate))
37 select curyear, curorder, lastorder, (curorder - lastorder) as difference
38   from thisyear
39   left outer join lastyear on curyear = lastyear + 1;
40
41 drop view if exists dbo.cust_aggregates;
42 go
43 create view dbo.cust_aggregates
44 as
45 select custid, year(orderdate) as year, count(orderid) as num_orders
46   from Sales.Orders
47   group by custid, year(orderdate);
48 go
49 select * from dbo.cust_aggregates order by custid;
50 go
51
52 drop view if exists dbo.my_products;
53 go
54 create view dbo.my_products
55 as
56 select p.productid, p.productname, p.unitprice, c.categoryname
57   from Production.Products p
```

```
58    join Production.Categories c
59    on p.categoryid = c.categoryid;
60  go
61  select * from dbo.my_products;
62  go
63
64  drop function if exists dbo.prods_by_month;
65  go
66  create function dbo.prods_by_month
67    (@prodid as int) returns table
68  as return
69  select month(o.orderdate) as month, od.productid, p.productname, sum(od.qty) as
        total_ordered
70    from Sales.Orders o
71    join Sales.OrderDetails od
72      on o.orderid = od.orderid
73    join Production.Products p
74       on p.productid = od.productid
75    where od.productid = @prodid
76    group by month(o.orderdate), od.productid, p.productname;
77  go
78  select * from dbo.prods_by_month(11)
79  go
```

Northwind queries

```
1   with USAcust as (select customerid from customers where country like 'USA')
2   select o.customerid, count(o.customerid) as cnt from orders o
3       where o.customerid in USAcust
4       group by o.customerid order by cnt desc;
5
6   with GERprod as (select s.supplierid, s.country, p.supplierid,  p.productid as pid,
7       p.productname from suppliers s join products p on s.supplierid = p.supplierid
8       where s.country like 'Germany'),
9   GERord as (select d.productid as pid, d.quantity, d.orderid, o.orderid, o.shipcountry
10      from orders o join order_details d on o.orderid = d.orderid
11      where o.shipcountry like 'Germany')
12      select distinct gp.productname,  sum(go.quantity) as TotalSold from GERprod gp join
            GERord go
13      on gp.pid = go.pid group by gp.productname order by TotalSold desc;
14
15  WITH EmployeeSubordinatesReport (EmployeeID, LastName, FirstName, NumberOfSubordinates,
        ReportsTo) AS
16  ( SELECT
17        EmployeeID, LastName, FirstName, (SELECT COUNT(1) FROM Employees e2
18            WHERE e2.ReportsTo = e.EmployeeID) as NumberOfSubordinates, ReportsTo
19    FROM Employees e)
20  SELECT Employee.LastName, Employee.FirstName, Employee.NumberOfSubordinates,
21     Manager.LastName as ManagerLastName, Manager.FirstName as ManagerFirstName
22  FROM EmployeeSubordinatesReport Employee
23     LEFT JOIN EmployeeSubordinatesReport Manager ON
24        Employee.ReportsTo = Manager.EmployeeID order by Employee.NumberOfSubordinates desc;
25
26  drop view if exists CustEmpPairs;
27  create view CustEmpPairs as
28        with CustEmpPairs (cid, eid) as (
29          select distinct customerid, employeeid from orders)
30          select c.customerid, c.companyname, c.contactname, e.employeeid, e.firstname, e.
                lastname
31          from customers c, employees e, CustEmpPairs o
32          where o.eid = e.employeeid and o.cid = c.customerid;
33  select * from CustEmpPairs where employeeid = 7;
34  select * from CustEmpPairs where customerid like 'CHOPS';
35  drop view if exists CustEmpPairs;
```