Advanced SQL Queries

Part 1. Subqueries (Select, From, and Where statements):

Tables: These are the tables that I will be working with.

SELECT \* FROM EmployeeSalary

+--------------------+---------------------------+--------------+

| EmployeeID | JobTitle | Salary |

+--------------------+---------------------------+--------------+

| 1001 | Salesman | 45000 |

| 1002 | Receptionist | 36000 |

| 1003 | Salesman | 63000 |

| 1004 | Accountant | 47000 |

| 1005 | HR | 50000 |

| 1006 | Regional Manager | 65000 |

| 1007 | Supplier Relations | 41000 |

| 1008 | Salesman | 48000 |

| 1009 | Accountant | 42000 |

| 1010 | NULL | 47000 |

| NULL | Salesman | 43000 |

+--------------------+---------------------------+--------------+

SELECT \* FROM EmployeeDemographics

+--------------------+-------------------+----------------+-----------+------------+

| EmployeeID | FirstName | LastName | Age | Gender |

+--------------------+-------------------+----------------+-----------+------------+

| 1001 | Jim | Halport | 30 | Male |

| 1002 | Pam | Beasely | 30 | Female |

| 1003 | Dwight | Schnute | 29 | Male |

| 1004 | Angela | Martin | 31 | Female |

| 1005 | Toby | Flenderson | 32 | Male |

| 1006 | Michael | Scott | 35 | Male |

| 1007 | Meredith | Palmer | 32 | Female |

| 1008 | Stanley | Hudson | 38 | Male |

| 1009 | Kevin | Malone | 31 | Male |

| 1011 | Ryan | Howard | 26 | Male |

| NULL | Holly | Flax | NULL | Female |

| 1013 | Darryl | Phibin | NULL | Male |

| 1011 | Ryan | Howard | 26 | Male |

| NULL | Holly | Flax | NULL | NULL |

+--------------------+-------------------+----------------+-----------+------------+

Subquery (Select):

SELECT EmployeeID, Salary, (SELECT AVG(Salary) FROM EmployeeSalary) as AllAvgSalary

FROM EmployeeSalary

+--------------------+--------------+--------------------------+

| EmployeeID | Salary | AllAvgSalary |

| 1001 | 45000 | 47909 |

| 1002 | 36000 | 47909 |

| 1003 | 63000 | 47909 |

| 1004 | 47000 | 47909 |

| 1005 | 50000 | 47909 |

| 1006 | 65000 | 47909 |

| 1007 | 41000 | 47909 |

| 1008 | 48000 | 47909 |

| 1009 | 42000 | 47909 |

| 1010 | 47000 | 47909 |

| NULL | 43000 | 47909 |

+--------------------+--------------+--------------------------+

The aforementioned query does the same as the following query.

Subquery(with Partition)

SELECT EmployeeID, Salary, AVG(Salary) OVER () as AllAvgSalary

FROM EmployeeSalary

+--------------------+--------------+--------------------------+

| EmployeeID | Salary | AllAvgSalary |

+--------------------+--------------+--------------------------+

| 1001 | 45000 | 47909 |

| 1002 | 36000 | 47909 |

| 1003 | 63000 | 47909 |

| 1004 | 47000 | 47909 |

| 1005 | 50000 | 47909 |

| 1006 | 65000 | 47909 |

| 1007 | 41000 | 47909 |

| 1008 | 48000 | 47909 |

| 1009 | 42000 | 47909 |

| 1010 | 47000 | 47909 |

| NULL | 43000 | 47909 |

+--------------------+--------------+--------------------------+

The following query will demonstrate why Group By doesn’t work.

Subquery(Group By):

SELECT EmployeeID, Salary, AVG(Salary) as AllAvgSalary

FROM EmployeeSalary

GROUP BY EmployeeID, Salary

ORDER BY 1, 2

+--------------------+--------------+--------------------------+

| EmployeeID | Salary | AllAvgSalary |

+--------------------+--------------+--------------------------+

| NULL | 43000 | 43000 |

| 1001 | 45000 | 45000 |

| 1002 | 36000 | 36000 |

| 1003 | 63000 | 63000 |

| 1004 | 47000 | 47000 |

| 1005 | 50000 | 50000 |

| 1006 | 65000 | 65000 |

| 1007 | 41000 | 41000 |

| 1008 | 48000 | 48000 |

| 1009 | 42000 | 42000 |

| 1010 | 47000 | 47000 |

+--------------------+--------------+--------------------------+

Since the query uses Group By which groups by both the ordered id and the Salary, we don’t get that all average salary (AllAvgSalary) that we are looking for.

Subquery(From):

SELECT a.EmployeeID, AllAvgSalary

FROM (SELECT EmployeeID, Salary, AVG(Salary) OVER () as AllAvgSalary

FROM EmployeeSalary) a

+--------------------+------------------------+

| EmployeeID | AllAvgSalary |

+--------------------+------------------------+

| 1001 | 47909 |

| 1002 | 47909 |

| 1003 | 47909 |

| 1004 | 47909 |

| 1005 | 47909 |

| 1006 | 47909 |

| 1007 | 47909 |

| 1008 | 47909 |

| 1009 | 47909 |

| 1010 | 47909 |

| NULL | 47909 |

+--------------------+------------------------+

The second Select query in the aforementioned query creates the EmployeeID/Salary/AllAvgSalary table as demonstrated above, but the first Select query allows me to specify specific columns of the second Select query. The example above demonstrates this.

Subquery(Where):

SELECT EmployeeID, JobTitle, Salary

FROM EmployeeSalary

WHERE EmployeeID IN (

SELECT EmployeeID

FROM EmployeeDemographics

WHERE Age > 30)

+--------------------+---------------------------+--------------+

| EmployeeID | JobTitle | Salary |

+--------------------+---------------------------+--------------+

| 1004 | Accountant | 47000 |

| 1005 | HR | 50000 |

| 1006 | Regional Manager | 65000 |

| 1007 | Supplier Relations | 41000 |

| 1008 | Salesman | 48000 |

| 1009 | Accountant | 42000 |

+--------------------+---------------------------+--------------+

Part 2. Temp Tables:

CREATE TABLE #temp\_Employee (

EmployeeID int,

JobTitle Varchar(100),

Salary int

)

SELECT \* FROM #temp\_Employee

+------------------+-------------+-----------+

| EmployeeID | JobTitle | Salary |

+------------------+-------------+-----------+

INSERT INTO #temp\_Employee VALUES (

‘1001’, ‘HR’, ‘45000’

)

SELECT \* FROM #temp\_Employee

+------------------+-------------+-----------+

| EmployeeID | JobTitle | Salary |

+------------------+-------------+-----------+

| 1001 | HR | 45000 |

+------------------+-------------+-----------+

INSERT INTO #temp\_Employee

SELECT \* FROM EmployeeSalary

+--------------------+---------------------------+-----------+

| EmployeeID | JobTitle | Salary |

+--------------------+---------------------------+-----------+

| 1001 | HR | 45000 |

| 1001 | Salesman | 45000 |

| 1002 | Receptionist | 36000 |

| 1003 | Salesman | 63000 |

| 1004 | Accountant | 47000 |

| 1005 | HR | 50000 |

| 1006 | Regional Manager | 65000 |

| 1007 | Supplier Relations | 41000 |

| 1008 | Salesman | 48000 |

| 1009 | Accountant | 42000 |

+--------------------+---------------------------+-----------+

DROP TABLE IF EXISTS #Temp\_Employee2

CREATE TABLE #Temp\_Employee2 (

JobTitle varchar(50),

EmployeesPerJob int,

AvgAge int,

AvgSalary int)

INSERT INTO #Temp\_Employee2

SELECT JobTitle, Count(JobTitle), Avg(Age), AVG(salary)

FROM EmployeeDemographics emp

JOIN EmployeeSalary sal

ON emp.EmployeeID = sal.EmployeeID

GROUP BY JobTitle

+---------------------------+------------------------+-------------+---------------+

| JobTitle | EmployeesPerJob | AvgAge | AvgSalary |

+---------------------------+------------------------+-------------+---------------+

| Accountant | 2 | 31 | 44500 |

| HR | 1 | 32 | 50000 |

| Receptionist | 1 | 30 | 36000 |

| Regional Manager | 1 | 35 | 65000 |

| Salesman | 3 | 32 | 52000 |

| Supplier Relations | 1 | 32 | 41000 |

+---------------------------+------------------------+-------------+---------------+

Part 3. String Functions (TRIM, LTRIM, RTRIM, Replace, Substring, Upper, Lower):

CREATE TABLE EmployeeErrors (

EmployeeID varchar(50),

FirstName varchar(50),

LastName varchar(50)

)

INSERT INTO EmployeeErrors VALUES

(‘1001 ’, ‘Jimbo’, ‘Halbert’),

(‘ 1002’, ‘Pamela’, ‘Beasely’),

(‘1005’, ‘TOby’, ‘Flenderson – Fired’)

SELECT \* FROM EmployeeErrors

+------------------+----------------+-----------------------+

| EmployeeID | FirstName | LastName |

+------------------+----------------+-----------------------+

| 1001 | Jimbo | Halbert |

| 1002 | Pamela | Beasely |

| 1005 | TOby | Flenderson – Fired |

+------------------+----------------+-----------------------+

The 1001 has 4 spaces after it, the 1002 has 4 spaces before it, and 1005 has neither.

String Functions (TRIM, LTRIM, RTRIM):

SELECT EmployeeID, TRIM(EmployeeID) as IDTRIM

FROM EmployeeErrors

+--------------------+---------------+

| EmployeeID | IDTRIM |

+--------------------+---------------+

| 1001 | 1001 |

| 1002 | 1002 |

| 1005 | 1005 |

+--------------------+---------------+

TRIM removed the 4 spaces after 1001, and the 4 spaces before 1002.

SELECT EmployeeID, LTRIM(EmployeeID) as IDTRIM

FROM EmployeeErrors

+--------------------+---------------+

| EmployeeID | IDTRIM |

+--------------------+---------------+

| 1001 | 1001 |

| 1002 | 1002 |

| 1005 | 1005 |

+--------------------+---------------+

LTRIM did not remove the 4 spaces after 1001, but did remove the 4 spaces before 1002.

SELECT EmployeeID, RTRIM(EmployeeID) as IDTRIM

FROM EmployeeErrors

+--------------------+---------------+

| EmployeeID | IDTRIM |

+--------------------+---------------+

| 1001 | 1001 |

| 1002 | 1002 |

| 1005 | 1005 |

+--------------------+---------------+

RTRIM removed the 4 spaces after 1001, but did not remove the 4 spaces before 1002.

String Functions (Replace):

SELECT LastName, REPLACE(LastName, ‘- Fired’, ‘’) as LastNameFixed

FROM EmployeeErrors

+------------------------+-----------------------+

| LastName | LastNameFixed |

+------------------------+-----------------------+

| Halbert | Halbert |

| Beasely | Beasely |

| Flenderson - Fired | Flenderson |

+------------------------+-----------------------+

The third record Flenderson – Fired was fixed to Flenderson.

String Functions (Substring):

SELECT SUBSTRING(FirstName, 1, 3)

FROM EmployeeErrors

+-------------------------+

| (No column name) |

+-------------------------+

| Jim |

| Pam |

| TOb |

+-------------------------+

SELECT err.FirstName, dem.FirstName

FROM EmployeeErrors err

JOIN EmployeeDemographics dem

ON err.FirstName = dem.FirstName

+------------------+------------------+

| FirstName | FirstName |

+------------------+------------------+

| TOby | Toby |

+------------------+------------------+

SELECT SUBSTRING(err.FirstName, 1, 3), SUBSTRING(dem.FirstName, 1, 3)

FROM EmployeeErrors err

JOIN EmployeeDemographics dem

ON SUBSTRING(err.FirstName, 1, 3) = SUBSTRING(dem.FirstName, 1, 3)

+----------------------------+----------------------------+

| (No column name) | (No column name) |

+----------------------------+----------------------------+

| Jim | Jim |

| Pam | Pam |

| TOb | Tob |

+-----------------------------+----------------------------+

SELECT err.FirstName, SUBSTRING(err.FirstName, 1, 3), dem.FirstName, SUBSTRING(dem.FirstName, 1, 3)

FROM EmployeeErrors err

JOIN EmployeeDemographics dem

ON SUBSTRING(err.FirstName, 1, 3) = SUBSTRING(dem.FirstName, 1, 3)

+---------------+------------------------+----------------+------------------------+

| FirstName | (No column name) | FirstName | (No column name) |

+---------------+------------------------+----------------+------------------------+

| Jimbo | Jim | Jim | Jim |

| Pamela | Pam | Pam | Pam |

| TOby | TOb | Toby | Tob |

+---------------+------------------------+-----------------+------------------------+

String Functions (UPPER and LOWER):

SELECT FirstName, LOWER(FirstName)

FROM EmployeeErrors

+------------------+----------------------------+

| FirstName | (No column name) |

+------------------+----------------------------+

| Jimbo | jimbo |

| Pamela | pamela |

| TOby | toby |

+-------------------+----------------------------+

LOWER changes the strings to lowercase.

SELECT FirstName, UPPER(FirstName)

FROM EmployeeErrors

+------------------+----------------------------+

| FirstName | (No column name) |

+------------------+----------------------------+

| Jimbo | JIMBO |

| Pamela | PAMELA |

| TOby | TOBY |

+-------------------+----------------------------+

UPPER changes the strings to uppercase.