**IS 441 SQL Practice (Subquery; OUTER JOIN; Self Join), 12/04/2015**

1. General: Places where subquery may be used

|  |  |  |
| --- | --- | --- |
|  | Position of Subquery in SELECT | Example |
| 1 | SELECT (Subquery) | SELECT RestaurantID, (SELECT AVG(SqFt) FROM …) |
| 2 | FROM (Subquery) | FROM (Select \* FROM… WHERE State=”CA”) |
| 3 | WHERE Boolean with (Subquery) | WHERE AnnualSales>(SELECT AVG(AnnualSales) FROM…) |
| 4 | GROUP BY |  |
| 5 | HAVING Boolean with (Subquery) | HAVING AVG(AnnualSales)> (SELECT AVG(AnnualSales) FROM…) |
| 6 | ORDER BY ; |  |

3 and 5 re logically the same – 3 is condition for filtering rows and 5 is condition for filtering groups.

1. Types of subqueries in terms of their positions in the SELECT-Clause
2. SELECT (Subquery)
   1. Purpose/occasion: the subquery results will be displayed; or: the desired results cannot be “legally” put in SELECT-clause unless it is a value returned by a subquery.
   2. Example: display individual annual sales (row value) and the average sales (set value/aggregate function value).
   3. Discussion: in this example,
      1. Must do it this way (why)?
      2. The “whole-table average”;
      3. The “by-city average” – involves passing parameters/arguments.
   4. Comment: canNOT be accomplished through JOIN or WHERE.
3. FROM (Subquery)
   1. Purpose/occasion: the subquery results (dynaset) will be used as the data source for further query.
   2. Example: display restaurant IDs and their cities for those restaurants in California.
   3. Discussion: in this example,
      1. The subset “restaurants in CA” can be obtained
         1. Through WHERE which filters all the rows using State=’CA’
         2. Through subquery which narrow all records down to those in CA.
   4. Comment:
      1. In this SPECIFIC example, it doesn’t have to use subquery, just showcase what subquery can do;
      2. In other cases (such as “WHERE EXIST” example in the text) there is no “normal” way in WHERE without subquery. 🡸 NOT in the final exam
4. WHERE Boolean with (Subquery): This is the most common and most “natural” type of subquery

Will NOT be in final exam

1. WHERE EXIST: This is more complex, and is correlated subquery
2. Results of Subqueries; Arguments/Parameters of Subqueries; Relationships between Main and SUBqueries
3. [Normally] The results of subqueries are not for display but for participation in a Boolean operation.
   1. Example: WHERE AnnualSales>(SELECT AVG(AnnualSales) FROM…)
   2. If results of a subquery are intended to be displayed, then the subquery must be placed in the SELECT-clause: SELECT RestaurantID, (SELECT AVG(SqFt) FROM …)
   3. The outer/main query and the subquery are “two different worlds” – meaning:
      1. They can use different sets of tables;
      2. They can involve different sets of fields;
      3. The fields in outer/main query is completely “alien” to those in the subquery, and vice versa;
      4. If we need to “force” the subquery to use the tables in the main/outer query, we need to pass parameters into the subquery
         1. Example: “List restaurants whose sales are higher than the AVG of sales in its own city”
4. OUTER JOIN
   1. Purpose/occasion: in two related tables, when we want to list all rows in one table (table “A”) no matter whether it has related rows in another (table “B”), that is when OUTER JOIN will be used.
   2. Syntax:

|  |  |
| --- | --- |
| … FROM A **LEFT** *[OUTER]* JOIN B | EVERY row in A, plus the matching B rows |
| Or |  |
| … FROM B **RIGHT** *[OUTER]* JOIN A | Matching rows in B, and **EVERY row in A** |

Note the comparison of LEFT and RIGHT, with the simultaneous SWITCHING of A & B.

* 1. Example:
     1. all **employee**s and their assigned **project**, even for those w/o a project
     2. all **customer**s and their order IDs, even for those who haven’t placed an **order**
     3. STUD LEFT JOIN CAR; or CAR RIGHT JOIN STUD – all students and the matching cars, including the students who do NOT own a car
  2. Reminder: do NOT forget the join condition!!!
     1. WHERE EMPLOYEE.ProjID = PROJECT.ProjID
     2. ON EMPLOYEE.ProjID = PROJECT.ProjID
     3. USING ProjID

1. Self Join
   1. Purpose/occasion: use one physical table (that is stored as “one piece”) for two logical tables or views.
   2. Key points:
      1. Define the two logical tables using alias;
      2. Identify the way (the field) the two views are related (PK=FK);
      3. State the join condition;
      4. Do NOT confuse the PK with the FK!
         1. “Employee’s manager ID is the same as Manager’s employee ID”
      5. Often with other conditions just as in a regular table join.
2. GROUP BY – things to watch
   1. Order of clause: first GROUP BY, then HAVING (very basic but…);
   2. Row value vs set value (aggregate function use);
   3. The values for Boolean operation in HAVING can be
      1. A value that has existed for the rows, where …
      2. A value that only exists for the whole group, where…

===============

Codes from practice:

[The first three questions are 2017; the rest are 2016]

**Example 1:**

Display the restaurants whose sales are greater than the average sale IN ITS OWN CITY.

Display also tTHAT average (the avg of the whole city where THAT restaurant is in) and give it an alias “CityAVG”

**“City AVG”:**



Dec5

|  |  |
| --- | --- |
| SELECT restaurantID, annualsales  FROM Restaurants  WHERE annualsales >  (SELECT AVG(annualsales)  FROM restaurants  GROUP BY City) | Road block here:  The subq returns M values, that WHERE is NOT allowed.  Our intention: the subq returns ONE value;  AND ON, specific value, that is –  THE avg of MY city.  But how do we assure that THE city inside the suq is THE city of the current restaurant’s city (as we are processing THE current restaurantID)?  Answer: Passing of parameter – THE city of the CURRENT restaurant is passed INSIDE the subq. |
| SELECT restaurantID, City, annualsales  FROM Restaurants Outside\_Rest  WHERE annualsales >  (SELECT AVG(annualsales)  FROM restaurants  GROUP BY City  HAVING city= Outside\_Rest.City  ) | Assures that:  1) only ONE avg will be returned; AND  2) THAT AVG is THE AVG of THE city that the current restaurant (in the outer query) is in |
|  |  |

**\*\*\* After the codes below that were created in 2015-2016, a new code segment created today (12/05/2017) is on the last page**

SELECT restaurantID, annualsales, City,

(SELECT AVG(annualsales) FROM restaurants

WHERE city = My.City

GROUP BY City)

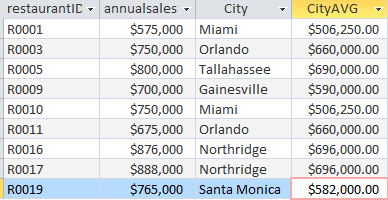
FROM Restaurants AS My

WHERE annualsales > #WHERE is the condition for rows

(SELECT AVG(annualsales) FROM restaurants

WHERE city = My.City

GROUP BY City)



**Example 2:**

Display the restaurants whose square footage is lower than the average SqFt IN ITS OWN TYPEofservice.

**“GangAVG”:**

SELECT restaurantID, squarefootage, TypeOfService,

(SELECT AVG(squarefootage) FROM restaurants

WHERE TypeOfService = MyGang.TypeOfService

GROUP BY TypeOfService) AS GangAVG

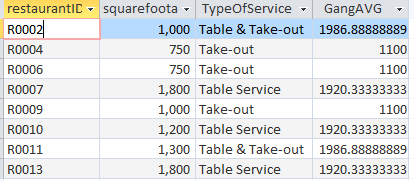
FROM Restaurants AS MyGang

WHERE squarefootage <

(SELECT AVG(squarefootage) FROM restaurants

WHERE TypeOfService = MyGang.TypeOfService

GROUP BY TypeOfService)



**Example 3:**

Display the restaurants whose sales are at least one and a half times the average of the restaurants

By the SAME Francisee. [Last name – from the Franchisees table]

**“HiInThePack”:**

SELECT restaurantID, annualsales, restaurants.franchiseeID, Lastname

(SELECT AVG(annualsales) FROM restaurants

WHERE franchiseeID = family.franchiseeID

GROUP BY franchiseeID) AS HiInThePack

FROM Restaurants AS family, Franchisees

WHERE family.franchiseeID = franchisees.franchiseeID

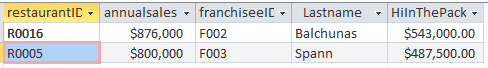
AND annualsales >=

1.5\*

(SELECT AVG(annualsales) FROM restaurants

WHERE franchiseeID = family.franchiseeID

GROUP BY franchiseeID)



Example 4: Self-Join

Attributing the number of orders from employees to their managers

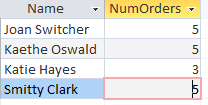
SELECT M.Name, COUNT(orderID)

FROM Employee M, Employee E, [order] O

WHERE E.eID=O.eID

AND E.ManagerID=M.eID

GROUP BY M.Name



**=====================================================**

**Below are codes from 2015/2016:**

SELECT restaurantID, city, annualsales, (select avg(annualsales) from restaurants where city=Maintable.city) AS TotalAvg

FROM Restaurants as MainTable

WHERE AnnualSales>(select avg(annualsales) from restaurants where city=Maintable.city)

order by city;

**“SELECT a subset from CA”:**

SELECT restaurantID, city

FROM (SELECT \* from Restaurants where state='ca');

Dec 5, 2017

HW#6, Prob 5:

SELECT I.itemid, description, orderdate

FROM [order] o, orderitem oi, inventory i

WHERE o.orderid=oi.orderid

AND oi.itemid=i.itemid

AND oi.itemid NOT IN 🡺 the action of exclusion, from the below:

(SELECT oi.itemid from orderitem oi, [order] o

WHERE o.orderid=oi.orderid those items that appeared in

AND MONTH(orderdate) IN (10,11,12) orders placed among months of 10,11,12

);

We need to completely exclude

THOSE items (i.e., itemIDs)

who appeared in orders that were placed among the months of 10,11,12

Prob #6: Emp who did not seel items in garden shop and during June/July

Idea: