

Exercise 7: Subqueries in Microsoft Access

The purpose of this handout is to illustrate the use of subqueries using SQL in order to pull information across tables.

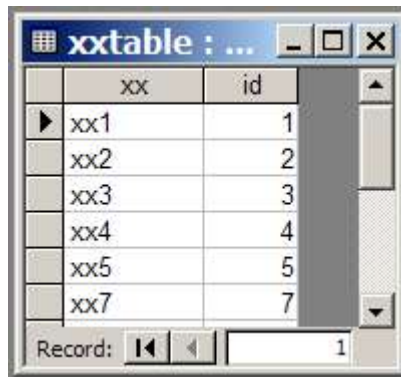
For this exercise, we will be using a database called “database_subqueries.mdb” located on the C:\ drive:

C:\Documents and Settings\student\My Documents\CIDER\database_subqueries.mdb

Example 1

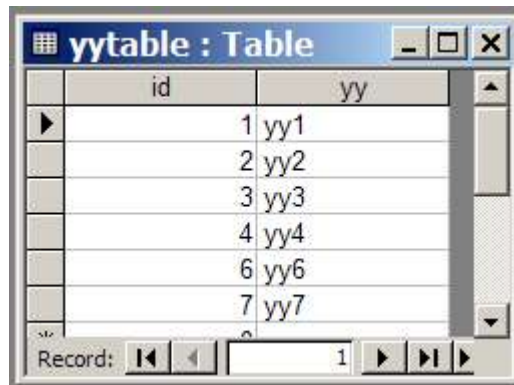
Tables

There are three tables for this example (the same tables used in the handout entitled “Joins in Action” that we worked with in the previous session). The first table is called “xxtable” and it has two fields: “xx” and “id”.



	xx	id
▶	xx1	1
	xx2	2
	xx3	3
	xx4	4
	xx5	5
	xx7	7

The second table is called “yytable” and it has two fields: “id” and “yy”.



	id	yy
▶	1	yy1
	2	yy2
	3	yy3
	4	yy4
	6	yy6
	7	yy7

The third table is called “zztable” and it has two fields: “id” and “zz”.

zztable : Table	
id	zz
1	zz1
2	zz2
3	zz3
4	zz4
5	zz5
6	zz6
7	zz7
8	zz8
9	zz9
10	zz10

Record: 1

Simple join with WHERE

```
SELECT zztable.id, zz
FROM xxtable, yytable, zztable
WHERE xxtable.id=yytable.id AND yytable.id=zztable.id;
```

Simple join with INNER JOIN

```
SELECT zztable.id, zz
FROM xxtable
INNER JOIN (yytable
            INNER JOIN zztable
                ON yytable.id = zztable.id)
ON xxtable.id = yytable.id;
```

Simple join with Subquery

```
SELECT zztable.id, zz
FROM zztable
WHERE zztable.id IN
    (SELECT xxtable.id
     FROM xxtable
     INNER JOIN yytable
         ON xxtable.id=yytable.id);
```

Simple RIGHT OUTER JOIN

```
SELECT yytable.id
```

```
FROM xxtable
RIGHT OUTER JOIN yytable
    ON xxtable.id=yytable.id
WHERE xx IS NULL;
```

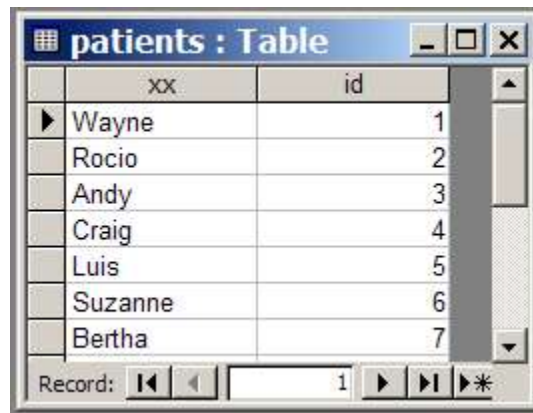
Simple RIGHT OUTER JOIN with Subquery

```
SELECT id, zz
FROM zztuple
WHERE id IN (SELECT yytable.id
            FROM xxtable
            RIGHT OUTER JOIN yytable
                ON xxtable.id=yytable.id
            WHERE xx IS NULL);
```

Example 2

Tables

For this example, we will use two different tables. The first table is called “patients” and it contains two fields: “xx”, the name of the first name of the patient, and “id”, each patient's identification number.



The screenshot shows a window titled "patients : Table" with a grid of data. The columns are labeled "xx" and "id". The rows contain the following data:

xx	id
Wayne	1
Rocio	2
Andy	3
Craig	4
Luis	5
Suzanne	6
Bertha	7

At the bottom of the window, there is a "Record:" label followed by navigation icons and the number "1".

The second table is called “visits” and contains data on the when each patient visited the clinic. The table has three fields: “visitid”, the visit identification number (the primary key); “id”, the patient identification number; and “zz”, the date of the visit to the clinic.

visitid	id	zz
1	1	4/4/2002
2	1	3/2/2003
3	2	7/10/2001
4	2	3/20/2000
5	3	4/8/1999
6	3	5/5/2000
7	3	6/5/2002
8	3	9/10/2003
9	4	1/2/2002
10	4	7/4/2003
11	5	4/5/2002
12	5	6/3/2003
13	6	10/5/2001
14	7	12/3/2002
* (AutoNumber)	0	

Record: 1

INNER JOIN patients with visits

```
SELECT xx, zz
FROM patients
INNER JOIN visits
    ON patients.id = visits.id;
```

Ordered INNER JOIN of patients with visits

```
SELECT xx, zz
FROM patients
INNER JOIN visits
    ON patients.id=visits.id
ORDER BY patients.id, visits.zz;
```

First Visit for Each Patient

```
SELECT result1.xx, MIN(result1.zz) AS "First Visit"
FROM (SELECT xx, zz
```

```

FROM patients
INNER JOIN visits
    ON patients.id=visits.id
ORDER BY patients.id, visits.zz) result1
GROUP BY result1.xx;

```

Note: in this SQL statement, we name the result of the inner query with an alias “result1”.

Last Visit for Each Patient

Similarly, we could determine the last visit for each patient by using the MAX() function:

```

SELECT result1.xx, MAX(result1.zz) AS "Last Visit"
FROM (SELECT xx, zz
      FROM patients
      INNER JOIN visits
          ON patients.id=visits.id
      ORDER BY patients.id, visits.zz) result1
GROUP BY result1.xx;

```

Again, we name the result of the inner query with an alias “result1”.

Count of Visits for Each Patient

```

SELECT id, COUNT(*) AS "Count of Visits"
FROM visits
GROUP BY id;

```

Remember the difference between COUNT(*) and COUNT(*expression*). I used the COUNT(*) function since I knew that there weren't any NULL values in the data.

However, now let's display the names of each patient instead of the identification number. In order to get the names, we will have to join the patients table with the visits table using the following SQL query:

```

SELECT result3.xx, COUNT(*) AS "Count of Visits"
FROM (SELECT xx, zz
      FROM patients
      INNER JOIN visits
          ON patients.id = visits.id) result3
GROUP BY result3.xx;

```

This time, our alias for the inner query is “result3”.

If you wanted to display the identification numbers along with the name, you would

modify the query to include the identification field (so that you can compare them to the results you did before):

```
SELECT result3.id, result3.xx, COUNT(*) AS "Count of Visits"
FROM (SELECT patients.id, xx, zz
      FROM patients
      INNER JOIN visits
            ON patients.id = visits.id) result3
GROUP BY result3.id, result3.xx;
```

Patients with Visits in 2000

```
SELECT xx, zz
FROM patients
INNER JOIN visits
      ON patients.id=visits.id
WHERE zz>=#1/1/2000# AND zz<=#12/31/2000#;
```

Count of Visits in 2000 by Patient

```
SELECT result4.xx, COUNT(*) AS "Count of Visits"
FROM (SELECT xx, zz
      FROM patients
      INNER JOIN visits
            ON patients.id=visits.id
            WHERE zz>=#1/1/2000# AND zz<=#12/31/2000#) result4
GROUP BY result4.xx;
```

The result of the inner query has the alias “result4”. This only lists the patients that have a visit in 2000 and counts the number of these visits for each patient. If we wanted to include identification numbers in the result, we would modify the query:

```
SELECT result4.id, result4.xx, COUNT(*) AS "Count of Visits"
FROM (SELECT patients.id, xx, zz
      FROM patients
      INNER JOIN visits
            ON patients.id=visits.id
            WHERE zz>=#1/1/2000# AND zz<=#12/31/2000#) result4
GROUP BY result4.id, result4.xx;
```

You might, however, want to display all patients in the result. To do this, we will LEFT JOIN the patients table with the result from the query we just did (which itself contains a subquery) as follows:

```

SELECT patients.xx, CountVisits
FROM patients
LEFT JOIN (SELECT result4.id, result4.xx, COUNT(*) AS CountVisits
          FROM (SELECT patients.id, xx, zz
                FROM patients
                INNER JOIN visits
                    ON patients.id=visits.id
                WHERE zz>=#1/1/2000#
                    AND zz<=#12/31/2000#) result4
          GROUP BY result4.id, result4.xx) result4b
ON patients.id=result4b.id;

```

Creating an Indicator for Patients with Visits in 2000

In many types of statistical analyses, one often needs to create an indicator field that signifies the presence or absence of some condition. For example,

```

SELECT patients.xx,
       YEAR(visits.zz) AS yearvisit,
       (YEAR(visits.zz)=2000) AS delta
FROM patients
INNER JOIN visits
    ON patients.id = visits.id;

```

Any logical expression can be put in the parentheses to create an indicator field (I named “delta”).

However, MS Access does not necessarily assign the values to 0 or 1 which is what we wanted. If you run the above query, you would get the following results:

	xx	yearvisit	delta
▶	Wayne	2002	0
	Wayne	2003	0
	Rocio	2001	0
	Rocio	2000	-1
	Andy	1999	0
	Andy	2000	-1
	Andy	2002	0
	Andy	2003	0
	Craig	2002	0
	Craig	2003	0
	Luis	2002	0
	Luis	2003	0
	Suzanne	2001	0
	Bertha	2002	0
*			

Record: 1 of 14

Thus, we need to make a slight modification of the SQL statement to accommodate this by using the absolute value function (ABS()):

```
SELECT patients.xx,
       YEAR(visits.zz) AS yearvisit,
       ABS((YEAR(visits.zz)=2000)) AS delta
FROM patients
INNER JOIN visits
      ON patients.id = visits.id;
```

Including Zeros in the Count of Patients with Visits in 2000

We can use this result in re-doing the count of patients with visits in 2000 to include a “zero” count for patients without any visits in 2000 (rather than blanks in the count column).

```
SELECT resulttble.xx,
       SUM(delta) AS "No Visits in 2000"
FROM (SELECT patients.xx,
            YEAR(visits.zz) AS yearvisit,
            ABS((YEAR(visits.zz)=2000)) AS delta
      FROM patients
      INNER JOIN visits
            ON patients.id = visits.id) resulttble
GROUP BY resulttble.xx;
```