I Didn't Know You Could Do that with SQL! *The Harder Stuff*

Skip Marchesani

Custom Systems Corp
Sparta, NJ 07871

973-579-1340
smarches@warwick.net

I Didn't Know You Could Do that with SQL: The Harder Stuff

Disclaimer:

This presentation may contain examples of code and names of companies or persons. The code is given for presentation purposes and has not been tested by IBM and/or Custom Systems Corp. Therefore IBM and/or Custom Systems Corp does not guarantee the reliability, serviceability, or function of the code and the code is provided "AS IS". IBM AND/OR CUSTOM SYSTEMS CORP EXPRESSLY DISCLAIMS ANY AND ALL WARRANTIES, INCLUDING BUT NOT LIMITED TO, THE WARRANTIES OF FITNESS FOR A PARTICULAR PURPOSE AND MERCHANTABILITY. Any names appearing in this presentation are designed to be fictitious and IBM and Custom Systems Corp makes no representations as to the accuracy of the names or data presented in accordance therewith.

Reproduction:

Overview

An Important Goal for this Presentation

- **An important goal for this presentation is to show you that SQL is just another programming language. The more you work and play with it, the more you realize the power of SQL and what it can do for you as an application development or database manipulation tool. With a little thought and creativity you will find you can use SQL for things that at first glance you did not think possible.**
- **This presentation is based on the SQL function available in V5R2 of OS/400, assumes you have a basic understanding of relational database concepts and SQL, and that you are familiar with using the SELECT, INSERT, UPDATE, and DELETE statements.**

Overview...

iSeries 400 Experts Journal

- **This presentation is based in large part on a series of articles that were published in *iSeries 400 Experts Journal* from September 2002 thru March 2004**
- ***iSeries 400 Experts Journal***
  - Published by iSeries 400 Experts (formerly the 400 Group)
  - www.iseries400experts.com
- **To subscribe and access these articles on the web contact:**
  Sarah J. Kimmel, Editor
  781-320-8909 ext 210
  sarahk@iseries400experts.com

Overview...

The Harder Stuff

- **Working with Edit Characters**
- **Basic Subselect**
- **Scalar Subselect**
- **Subselect and INSERT, UPDATE, & DELETE**
- **Subselect and CREATE TABLE**
- **Subselect and Derived Table**
- **Identifying Potential Duplicate Rows**
- **Search by Sound with SOUNDEX**
- **Summary**
- **V5R2 SQL Information Sources**

SQL Syntax Used in Examples

- **All UPPER CASE - SQL required syntax**
- **All lower case - SQL parameter data supplied by end user**
- **One or more characters enclosed in single quotes ('S') is a literal used for comparison purposes and is supplied by the end user.**
- **Anything enclosed in [ ] is optional SQL syntax or optional parameter data**

Working with Edit Characters

Working with Edit Characters

Transferring data to/from iSeries or AS/400

- **When data is transferred *from* a client PC to an iSeries (AS/400) it could contain edit characters that may not be compatible with the DB2 UDB (DB2/400) data format.**
- **When data is transferred *to* a client PC from an iSeries (AS/400) it might lack edit characters that may be required by the PC end user.**
- **Examples:**
  - Telephone number: 973-555-1212
  - Social Security number: 132-54-7698
  - Date: 12/25/03
  - Time: 10:30 am

Working with Edit Characters...

The Question

- **Can SQL be used to *easily* remove extraneous edit characters or insert required edit characters.**

The Answer

- **With a little thought and some creativity the answer is *YES!***

Case Study

PC Name and Address File

- **Excel spreadsheet**
- **Character (alpha) phone number**
  - Format is Char(12) AAA-EEE-NNNN
    - AAA = Area Code
    - EEE = Phone Exchange
    - NNNN = Phone Number
  - Hyphens for editing included as part of the data
- **PC file to be imported into DB2 UDB database**

Case Study...

PC File to be Imported into DB2 UDB Database

- **Name and Address Table - NAMEMSTR**
- **Decimal (numeric) phone number - PHONE**
  - Format is Dec(10,0) AAAEEENNNN
    - AAA = Area Code
    - EEE = Phone Exchange
    - NNNN = Phone Number
  - Hyphens are not valid in a numeric column

Case Study...

PC File Layout

- **IDNBR dec**
- **FNAME char**
- **LNAME char**
- **COMPNY char**
- **ADDRL1 char**
- **ADDRL2 char**
- **CITY char**
- **STATE char**
- **ZIP char**
- **PHONEA char**
  AAA-EEE-NNNN

DB2 UDB Table Format

- **IDNBR dec**
- **FNAME char**
- **LNAME char**
- **COMPNY char**
- **ADDRL1 char**
- **ADDRL2 char**
- **CITY char**
- **STATE char**
- **ZIP char**
- **PHONE dec**
  AAAEEENNNN
- **Other columns**

Case Study...

The Question:

- **How can SQL be used to remove the hyphens in the character phone number and then change it to a decimal data type?**

Removing the Hyphens

Import the PC File

- **Use Excel *Transfer Data to AS/400* to import PC file into a work table in DB2 UDB**
  - Assign import file name as **NAMEWORK**
  - Assign phone number column name as **PHONEA**
  - Verify that phone number is defined as Char(12)

Note: ***Transfer Data To/From AS/400*** is an Excel plug-in that is part of the licensed product for File Transfer within Client Access Express. When installing Client Access Express, you must select the option to install File Transfer to have the Excel plug-in installed.

Removing the Hyphens...

Import the PC File and Add PHONE Column

- **Add second phone number column with PHONE as name and data type of Dec(10,0) to match same column in NAMEMSTR**

```
ALTER  TABLE  Namework
   ADD  COLUMN  phone  DEC (10 , 0)
     NOT NULL WITH DEFAULT
```

Removing the Hyphens...

Format of PHONEA is Char(12) AAA-EEE-NNNN

- **Area code is in positions 1 thru 3**
- **1st hyphen is in position 4**
- **Phone exchange is in positions 5 thru 7**
- **2nd hyphen is position 8**
- **Phone number is in positions 9 thru 12**

Removing the Hyphens...

Update PHONE in NAMEWORK

- **Update the column PHONE by removing the hyphens from PHONEA and then changing its data type to Dec(10,0)**

```
UPDATE namework SET PHONE =
   CAST(CONCAT(CONCAT(SUBSTR(phonea,1,3),
         SUBSTR(phonea,5,3)),
         SUBSTR(phonea,9,4)) AS DEC(10,0))
```

Removing the Hyphens...

CONCAT Builds PHONE w/o Hyphens - Dec(10,0)

- **Join two character strings into a single string**
- **Right (or 2nd) CONCAT**
  - AAA + EEE = AAAEEE
  - 123 + 567
- **Left (or 1st) CONCAT**
  - AAAEEE + NNNN = **AAAEEENNNN**
  - 123567 + 9101112

CAST Function

- **Changes Char(12) to Dec(10,0)**
  - Assuming valid numeric characters in the column

Removing the Hyphens...

Insert/Add NAMEWORK Rows into NAMEMSTR

- **PHONE exists as a 10 position numeric field with no hyphens**
- **NAMEWORK rows can be inserted into NAMEMSTR using a SUBSELECT**
  - More on SUBSELECT later

```
INSERT INTO namemstr (idnbr, fname, lname,
       compny, addrl1, addrl2, city, state,
       zip, contry, phone)
  SELECT idnbr, fname, lname, compny,
         addrl1, addrl2, city, state, zip,
         contry, phone
       FROM namework
```

Removing the Hyphens...

Considerations for Inserting NAMEWORK Rows

- **PHONEA not included when inserting rows from NAMEWORK into NAMEMSTR**
- **Column names in INTO clause and SUBSELECT must be listed in the same position or sequence**
  - One for one, positional relationship between columns in the INTO clause and the columns in the SUBSELECT
  - Column names do not have to be the same but column attributes (data type) must be compatible
- **For name and address and possibly other table types, and prior to doing insert, consider checking NAMEWORK for duplicates that already exist in NAMEMSTR**
  - More on this later

Removing the Hyphens...

Can It be Done as a One Step Process?

- **YES - Include the CONCAT, SUBSTR, and CAST of PHONEA as part of the SUBSELECT in the INSERT STATEMENT**

```
INSERT INTO namemstr (idnbr, fname, lname,
            compny, addrl1, addrl2, city, state, zip,
            contry, phone)
   SELECT idnbr, fname, lname, compny, addrl1,
              addrl2, city, state, zip, contry,
              CAST(CONCAT(CONCAT(SUBSTR(phonea,1,3),
                    SUBSTR(phonea,5,3)),
                     SUBSTR(phonea,9,4)) AS DEC(10,0))
          FROM namework
```


Inserting Hyphens for Readability

The Next Question:

- **How can SQL be used to insert hyphens in the numeric phone number so the DB2 UDB table can be exported to an Excel file on the PC?**

Inserting Hyphens for Readability...

Format of PHONE is Dec(10,0) AAAEEENNNN

- **Area code is in positions 1 thru 3**
- **Phone exchange is in positions 4 thru 6**
- **Phone number is in positions 7 thru 10**

Inserting Hyphens for Readability...

Create Export File Using Interactive SQL - STRSQL

- **SELECT required columns and use CONCAT, SUBSTR, and CAST to change PHONE data type to Char(10) and insert hyphens in the appropriate positions, and create PHONEA**
- **Use F13 then Option 1 to direct SELECT output to a file.**

```
SELECT  idnbr,  fname,  lname,  compny,  addrl1,
                addrl2,  city,  state,  zip,  contry,
     CONCAT(CONCAT(CONCAT(CONCAT(
                SUBSTR(CAST(phone  AS  CHAR(10)),1,3), '-'),
           SUBSTR(CAST(phone  AS  CHAR(10)),4,3)),  '-'),
      SUBSTR(CAST(phone  AS  CHAR(10)),7,4))  AS phonea
FROM  namemstr
```

Inserting Hyphens for Readability...

SUBSTR(CAST(PHONE AS CHAR(10)),start,length)

- **PHONE is a numeric field - Dec(10,0), and only a character field can be operated on by CONCAT and SUBSTR.**
- **PHONE must be CAST or converted to a character field each time it is operated on by CONCAT or SUBSTR**

Inserting Hyphens for Readability...

CONCAT Builds PHONEA with Hyphens - Char(12)

- **Right outer (or 4th) CONCAT**
  - SUBSTR(CAST(**PHONE** AS CHAR(10)),1,3), '-'),
  - AAA-
- **Right inner (or 3rd) CONCAT**
  - SUBSTR(CAST(**PHONE** AS CHAR(10)),4,3)),
  - AAA-EEE
- **Left inner (or 2nd) CONCAT**
  - '-'),
  - AAA-EEE-
- **Left outer (or 1st) CONCAT**
  - SUBSTR(CAST(**PHONE** AS CHAR(10)),7,4)
  - **AAA-EEE-NNNN**

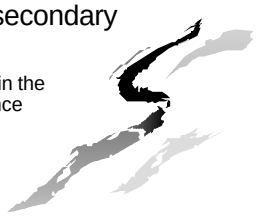
Basic Subselect

Basic Subselect

What is a Subselect (aka Subquery)?

- **Capability to nest 1 to 31 SELECT statements inside a SELECT, INSERT, UPDATE, or DELETE statement**
- **Provides significantly enhanced function**
- **Can reference two or more tables or views within a single, compound SQL statement without having to do a join of the tables or views involved**
  - Similar to RPG pgm that reads primary file, chains to secondary file, and retrieves or updates rows in primary based on secondary

Note: The term subquery and subselect are often used interchangeably - even in the IBM documentation! Since the difference is subtle (we won't go into the difference here), and to simplify discussion, only the term subselect will be used in this presentation.



Sample Tables - Subselect Examples

- **Employee Table - EMP**

NBR	NAM	CLS	SEX	DPT	SAL
10	Ed	5	M	911	7,000
20	Heikki	2	M	901	6,000
30	John	5	M	977	3,200
40	Mike	4	M	977	6,500
50	Marcela	3	F	911	7,500
60	Frank	2	M	990	6,500

- **Department Table - DEP**

DPT	DNM
901	Accounts
977	Manufact
911	Sales
990	Spares

SELECT with Simple Subselect

Using Subselect to Build a Selection List

- **SELECT all employees that work in a department that has a name beginning with 'S' (upper case S)**
- **DNM - Department name is only in DEP (Department Master Table) and is not in EMP (Employee Master Table)**
- **Is there an easy way to do this without doing an inner join between EMP and DEP?**

YES - There Is!

SELECT with Simple Subselect...

Using Subselect to Build a Selection List cont.

- ***Yes!*** **- Subselect is the solution - JOIN not required**

```
SELECT nbr, nam, dpt, sal FROM emp
      WHERE dpt IN
            (SELECT dpt FROM dep WHERE dnm LIKE 'S%')
      ORDER BY nbr
```

NBR	NAM	DPT	SAL
10	Ed	911	7,000
50	Marcela	911	7,500
60	Frank	990	6,500

SELECT with Simple Subselect...

Subselect Terminology

- **First or left most SELECT is the primary or outer select**

```
SELECT nbr, nam, dpt, sal FROM emp
      WHERE dpt IN
            (SELECT dpt FROM dep WHERE dnm LIKE 'S%')
      ORDER BY nbr
```

- **Second or right most SELECT is the subselect and is also called the inner select**

```
SELECT nbr, nam, dpt, sal FROM emp
      WHERE dpt IN
            (SELECT dpt FROM dep WHERE dnm LIKE 'S%')
      ORDER BY nbr
```

Using Subselect with IN Predicate

Using the IN Predicate

- **Compares a column value in the WHERE clause on the left side of the IN predicate with a set or list of values on the right side of the IN predicate**
- **Subselect provides the list of values for the IN predicate**

```
WHERE  dpt  IN
       (SELECT  dpt  FROM  dep  WHERE  dnm  LIKE  'S%')
```

- When subselect is used on the right side of the IN predicate, only ***single*** column is allowed in the column list for the SELECT
  - **dpt**
- Multiple rows can be returned by subselect
  - Builds selection list

Using Subselect with IN Predicate...

How Does the IN Predicate Work with Subselect?

- **SQL resolves (evaluates and executes) the subselect first**

```
SELECT  nbr,  nam,  dpt,  sal  FROM  emp
     WHERE  dpt  IN
          (SELECT  dpt  FROM  dep  WHERE  dnm  LIKE  'S%')
     ORDER  BY  nbr
```

DEP

DPT	DNM
901	Accounts
977	Manufact
911	**Sales**
990	**Spares**

Using Subselect with IN Predicate...

How Does the IN Predicate Work with Subselect?...

- **Subselect retrieves all rows that satisfy selection criteria in WHERE clause**
- **Retrieved rows become selection list for IN predicate when SQL executes outer select**

```
SELECT nbr, nam, dpt, sal FROM emp
    WHERE dpt IN (911, 990)
    ORDER BY nbr
```

NBR	NAM	DPT	SAL
10	Ed	911	7,000
50	Marcela	911	7,500
60	Frank	990	6,500

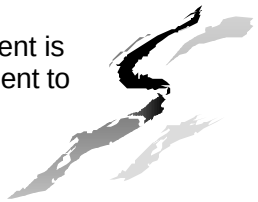
DEP

DPT	DNM
901	Accounts
977	Manufact
911	**Sales**
990	**Spares**

Basic Subselect - Quick Recap

Maximum of 31 subselects per SQL statement

- **Also can be referred to as inner selects**
- **Nest within**
  - SELECT
  - INSERT
  - UPDATE
  - DELETE
  - CREATE TABLE - V5R2
- **Performance**
  - As the number of subselects within an SQL statement is increased, the longer it will take for that SQL statement to execute



Scalar Subselect

Scalar Subselect

What is a Scalar Subselect?

- **A Scalar Subselect is used in place of an SQL expression or function**

```
SELECT fld1, fld2, (SELECT flda FROM tablea...)
   FROM table1...

UPDATE table1
   SET fld1 = SELECT flda FROM tablea...
```

Scalar Subselect...

Example

- **SELECT all employees that work in a department that has a name beginning with 'S' (upper case S) and include the department name in the result set**
- **DNM - Department name is only in DEP (Department Master Table) and is not in EMP (Employee Master Table)**
- **Is there an easy way to do this without doing an inner join between EMP and DEP?**
- **YES - There Is!**

Scalar Subselect...

Example cont.

- ***Yes!* - Scalar Subselect is the solution - JOIN not required**

```
SELECT nbr, nam, dpt, sal,
        (SELECT dnm FROM dep b
            WHERE a.dpt = b.dpt) AS dept_name
    FROM emp a
    WHERE dpt IN
        (SELECT dpt FROM dep WHERE dnm LIKE 'S%')
    ORDER BY nbr
```

NBR	NAM	DPT	SAL	DEPT NAME
10	Ed	911	7,000	Sales
50	Marcela	911	7,500	Sales
60	Frank	990	6,500	Spares

DEP

DPT	DNM
901	Accounts
977	Manufact
911	**Sales**
990	**Spares**

Scalar Subselect...

What's this DEP B and EMP A Stuff?

- **Correlation Names**
  - Alternate name for a table or view referenced in an SQL statement
  - SQL supports long names - up to 128 characters
  - Temporary short name used as an alternate to the long name
- **Correlation name used to qualify column name**
  - TABLE-NAME.COLUMN-NAME
  - CORRELATION-NAME.COLUMN-NAME
- **When the same column name is used in two tables (or views), and there is a reference to that column name for each of these tables in an SQL expression, that column name must be qualified to resolve to the correct table**

Scalar Subselect...

What's this DEP B and EMP A Stuff...

- **Correlation Name Assigned in FROM Clause**

```
SELECT nbr, nam, dpt, sal,
       (SELECT dnm FROM dep b
            WHERE a.dpt = b.dpt) AS dept_name
  FROM emp a
  WHERE dpt IN
       (SELECT dpt FROM dep WHERE dnm LIKE 'S%')
  ORDER BY nbr
```

NBR	NAM	DPT	SAL	DEPT NAME
10	Ed	911	7,000	Sales
50	Marcela	911	7,500	Sales
60	Frank	990	6,500	Spares

DEP

DPT	DNM
901	Accounts
977	Manufact
911	**Sales**
990	**Spares**

Subselect and INSERT, UPDATE, and DELETE

Subselect and INSERT

Using Subselect in the INSERT Statement

- **Create a workfile and populate with rows and columns selected from EMP**

```
CREATE TABLE   empname
          (number   DEC (3,0)   NOT NULL WITH DEFAULT,
           name   CHAR (10)   NOT NULL WITH DEFAULT,
           dept   DEC (3,0)   NOT NULL WITH DEFAULT)
```

```
INSERT INTO   empname   (number,   name,   dept)
    SELECT   nbr,   name,   dpt   FROM   emp
```

NBR	NAM	DPT
10	Ed	911
20	Heikki	901
30	John	977
40	Mike	977
50	Marcela	911
60	Frank	990

Subselect and INSERT...

Using Subselect in the INSERT Statement

- **Create a workfile and populate with rows and columns selected from EMP and DEP**

```
INSERT INTO empname (number, name, dept, dptnam)
    SELECT nbr, nam, dpt,
        (SELECT dnm FROM dep b
            WHERE a.dpt = b.dpt) AS dptnam
    FROM emp a
```

NBR	NAM	DPT	DPTNAM
10	Ed	911	Sales
20	Heikki	901	Accounts
30	John	977	Manufact
40	Mike	977	Manufact
50	Marcela	911	Sales
60	Frank	990	Spares

Subselect and DELETE

Using Subselect in the DELETE Statement

- **Delete rows for employees that have been transferred to another division**

```
DELETE FROM emp WHERE nbr IN
    (SELECT nbr FROM transfer)
```

EMP

NBR	NAM	CLS	SEX	DPT	SAL
10	Ed	5	M	911	7,000
20	Heikki	2	M	901	6,000
30	John	5	M	977	3,200
40	Mike	4	M	977	6,500
50	Marcela	3	F	911	7,500
60	Frank	2	M	990	6,500

TRANSFER

NBR	NAM	DPT
20	Heikki	901
30	John	977
50	Marcela	911

Subselect and DELETE...

Using Subselect in the DELETE Statement cont.

EMP - Before Delete

NBR	NAM	CLS	SEX	DPT	SAL
10	Ed	5	M	911	7,000
20	Heikki	2	M	901	6,000
30	John	5	M	977	3,200
40	Mike	4	M	977	6,500
50	Marcela	3	F	911	7,500
60	Frank	2	M	990	6,500

TRANSFER

NBR	NAM	DPT
20	Heikki	901
30	John	977
50	Marcela	911

EMP - After Delete

NBR	NAM	CLS	SEX	DPT	SAL
10	Ed	5	M	911	7,000
40	Mike	4	M	977	6,500
60	Frank	2	M	990	6,500

Subselect and UPDATE

Simple Scalar Subselect in the UPDATE Statement

- **Update each employee's salary to the amount listed in the new salary table**

```
UPDATE emp aa
    SET sal = (SELECT nsal FROM newsal1 bb
                  WHERE aa.nbr = bb.nbr)
```

EMP

NBR	NAM	CLS	SEX	DPT	SAL
10	Ed	5	M	911	7,000
20	Heikki	2	M	901	6,000
30	John	5	M	977	3,200
40	Mike	4	M	977	6,500
50	Marcela	3	F	911	7,500
60	Frank	2	M	990	6,500

NEWSAL1

NBR	NSAL
10	7,890
20	6,890
30	4,090
40	7,390
50	8,390
60	7,390

Subselect and UPDATE...

Simple Scalar Subselect in the UPDATE Statement cont.

EMP - Before Update

NBR	NAM	CLS	SEX	DPT	SAL
10	Ed	5	M	911	7,000
20	Heikki	2	M	901	6,000
30	John	5	M	977	3,200
40	Mike	4	M	977	6,500
50	Marcela	3	F	911	7,500
60	Frank	2	M	990	6,500

NEWSAL1

NBR	NSAL
10	7,890
20	6,890
30	4,090
40	7,390
50	8,390
60	7,390

EMP - After Update

NBR	NAM	CLS	SEX	DPT	SAL
10	Ed	5	M	911	**7,890**
20	Heikki	2	M	901	**6,890**
30	John	5	M	977	**4,090**
40	Mike	4	M	977	**7,390**
50	Marcela	3	F	911	**8,390**
60	Frank	2	M	990	**7,390**

Each row in EMP must have a corresponding row in NEWSAL1

Subselect and UPDATE...

'Null Values Not Allowed' Error

- **If each row in EMP *<u>does not</u>* have a corresponding row in NEWSAL1**

```
UPDATE emp aa
    SET sal = (SELECT nsal FROM newsal2 bb
                  WHERE aa.nbr = bb.nbr)
```

EMP

NBR	NAM	CLS	SEX	DPT	SAL
10	Ed	5	M	911	7,000
20	Heikki	2	M	901	6,000
30	John	5	M	977	3,200
40	Mike	4	M	977	6,500
50	Marcela	3	F	911	7,500
60	Frank	2	M	990	6,500

NEWSAL2

NBR	NSAL
10	7,520
20	6,750
30	3,720
60	7,020

Error Msg: *Null values not allowed in column or variable SAL* - **WHY?**

Subselect and UPDATE...

'Null Values Not Allowed' Error - Why?

EMP - Before Update

NBR	NAM	CLS	SEX	DPT	SAL
10	Ed	5	M	911	7,000
20	Heikki	2	M	901	6,000
30	John	5	M	977	3,200
40	Mike	4	M	977	6,500
50	Marcela	3	F	911	7,500
60	Frank	2	M	990	6,500

NEWSAL2

NBR	NSAL
10	7,520
20	6,750
30	3,720
60	7,020

EMP - After Update (assumes no commitment control)

NBR	NAM	CLS	SEX	DPT	SAL
10	Ed	5	M	911	**7,520**
20	Heikki	2	M	901	**6,750**
30	John	5	M	977	**3,720**
40	Mike	4	M	977	6,500
50	Marcela	3	F	911	7,500
60	**Frank**	**2**	**M**	**990**	**6,500**

Employees 10, 20, and 30 are updated based on NEWSAL2. When employee 40 read, there is no corresponding NSAL value in NEWSAL2. Therefore SQL attempts to change SAL to a null value which is not allowed. Update process ends with error at that point and employees 50 and 60 are not processed.

Subselect and UPDATE...

Avoiding the *'Null Values Not Allowed'* Error

- **Add WHERE clause with second subselect to UPDATE**

```
UPDATE emp aa
    SET sal = (SELECT nsal FROM newsal2 bb
                  WHERE aa.nbr = bb.nbr)
    WHERE aa.nbr IN (SELECT nbr FROM newsal2)
```

EMP

NBR	NAM	CLS	SEX	DPT	SAL
10	Ed	5	M	911	7,000
20	Heikki	2	M	901	6,000
30	John	5	M	977	3,200
40	Mike	4	M	977	6,500
50	Marcela	3	F	911	7,500
60	Frank	2	M	990	6,500

NEWSAL2

NBR	NSAL
10	7,520
20	6,750
30	3,720
60	7,020

Subselect and UPDATE...

Avoiding the *'Null Values Not Allowed'* Error...

EMP - Before Update

NBR	NAM	CLS	SEX	DPT	SAL
10	Ed	5	M	911	7,000
20	Heikki	2	M	901	6,000
30	John	5	M	977	3,200
40	Mike	4	M	977	6,500
50	Marcela	3	F	911	7,500
60	Frank	2	M	990	6,500

NEWSAL2

NBR	NSAL
10	7,520
20	6,750
30	3,720
60	7,020

EMP - After Update

NBR	NAM	CLS	SEX	DPT	SAL
10	Ed	5	M	911	**7,520**
20	Heikki	2	M	901	**6,750**
30	John	5	M	977	**3,720**
40	Mike	4	M	977	6,500
50	Marcela	3	F	911	7,500
60	Frank	2	M	990	**7,020**

Subselect and CREATE TABLE

Subselect and CREATE TABLE

V5R2 Scalar Subselect

```
CREATE TABLE workfile AS
      (SELECT a.dpt, SUM(sal) AS totsal,
            (SELECT dnm FROM dep b
               WHERE a.dpt = b.dpt) AS dnm
         FROM emp a
         GROUP BY a.dpt, 3)
   WITH DATA
```

Dept	Total Salary	Dept Name
901	6,000	Accounts
911	14,500	Sales
977	9,700	Manufact
990	6,500	Spares

Subselect and CREATE TABLE...

CREATE TABLE Syntax Using V5R2 Scalar Subselect

- **Each derived column defined in the scalar subselect within the CREATE TABLE statement *must* be given a name using the AS operator**
- **CREATE TABLE statement *must* be ended with one of the following two clauses**
  - **WITH DATA**
    - Table is populated with rows and columns that match table definition and selection criterion
  - **WITH NO DATA**
    - Table is created as empty table with no rows

Subselect and CREATE TABLE...

CREATE TABLE and V5R2 Scalar Subselect

- **Provides SQL with missing native functions**
  - CPYF - with or without data
  - CRTDUPOBJ - with or without data
  - Field Reference File support

CREATE TABLE table_name AS (SELECT...)...

- ***Subset*** of fields in reference table used in created table

CREATE TABLE table_name LIKE reference_table_name...

- ***All*** fields in reference table used in created table

- ***One Stop Shopping*** **- create table and populate with data**
  - Not available with native interface!

Subselect and Derived Table

Subselect and Derived Table

What is a Derived a Table?

- **A derived table is created in the FROM clause of a SELECT statement as part of the statement execution process**
- **Subselect can be used to derive a table in the FROM clause of a SELECT statement**
- **Example - SELECT the first row of a group of rows where SELECT DISTINCT or GROUP BY cannot be used to satisfy the selection criteria**

Subselect and Derived Table...

SELECT the First Row of a Group...

- **What is the first or lowest employee number in each department?**

EMP

NBR	NAM	CLS	SEX	DPT	SAL
10	**Ed**	**5**	**M**	**911**	**7,000**
20	**Heikki**	**2**	**M**	**901**	**6,000**
30	**John**	**5**	**M**	**977**	**3,200**
40	Mike	4	M	977	6,500
50	Marcela	3	F	911	7,500
60	**Frank**	**2**	**M**	**990**	**6,500**

- **SELECT DISTINCT or GROUP BY cannot be used to satisfy this selection criteria**

Subselect and Derived Table...

SELECT the First Row of a Group...

- **What is the first employee number in each department?**

```
SELECT * FROM emp WHERE nbr IN
    (SELECT number FROM
        (SELECT dpt, MIN(nbr) AS number
            FROM emp
            GROUP BY dpt)
        AS first-row-table)
    ORDER BY dpt
```

Subselect and Derived Table...

SELECT the First Row of a Group...

- **What is the first employee number in each department?**

EMP

NBR	NAM	CLS	SEX	DPT	SAL
10	**Ed**	**5**	**M**	**911**	**7,000**
20	**Heikki**	**2**	**M**	**901**	**6,000**
30	**John**	**5**	**M**	**977**	**3,200**
40	Mike	4	M	977	6,500
50	Marcela	3	F	911	7,500
60	**Frank**	**2**	**M**	**990**	**6,500**

Result Set for SELECT Statement

NBR	NAM	CLS	SEX	DPT	SAL
20	Heikki	2	M	**901**	**6,000**
10	Ed	5	M	**911**	**7,000**
30	John	5	M	**977**	**3,200**
60	Frank	2	M	**990**	**6,500**

Identifying Potential Duplicate Rows

Identifying Potential Duplicate Rows

Inner Join for Identifying Potential Duplicate Rows

- **Duplicate rows in a table can be a significant waste of time, money, and other resources - especially if the table contains contact or name and address information**
- **Inner join used because a resulting row is returned when a row in the inner (right) table matches a row in the outer (left) table**

Duplicate Row Case Study

Mailing List Table

- **Table Name:** **MAILLIST**
- **Total Rows:** **25,000+**
- **Duplicate Rows:** **Many - visual verification**
- **Row Format:**

Column Name	Length	Data Type	Description
IDNBR	5.0	Dec	Unique ID Number
FNAME	15	Char	First Name
LNAME	30	Char	Last Name
COMPNY	50	Char	Company Name
ADDRL1	50	Char	Address Line 1
ADDRL2	50	Char	Address Line 2
CITY	25	Char	City
STATE	2	Char	State
ZIP	10	Char	Zip Code

Duplicate Row Case Study...

Creating the Inner Join - Version 1

```
SELECT a.idnbr, a.fname, a.lname, a.compny, a.addrl1,
       a.addrl2, a.city, a.state, a.zip
  FROM maillist a INNER JOIN maillist b
    ON a.fname = b.fname
      AND a.lname = b.lname
      AND a.compny = b.compny
      AND a.addrl1 = b.addrl1
      AND a.addrl2 = b.addrl2
      AND a.city = b.city
      AND a.state = b.state
      AND a.zip = b.zip
      AND a.idnbr < > b.idnbr
ORDER BY a.lname, a.compny a.fname
```

- **No rows returned - Why?**
  - Inconsistencies with address and other data

Duplicate Row Case Study...

Creating the Inner Join - Version 2

```
SELECT a.idnbr, a.fname, a.lname, a.compny, a.city
    FROM maillist a INNER JOIN maillist b
      ON a.state = b.state
        AND SUBSTR(a.fname,1,3) = SUBSTR(b.fname,1,3)
        AND SUBSTR(a.lname,1,3) = SUBSTR(b.lname,1,3)
        AND SUBSTR(a.compny,1,3) = SUBSTR(b.compny,1,3)
        AND SUBSTR(a.city,1,3) = SUBSTR(b.city,1,3)
        AND a.idnbr < > b.idnbr
  ORDER BY a.lname, a.compny, a.fname
```

- **Four rows returned**

Duplicate Row Case Study...

Four Rows Returned

- **Which rows are duplicates, which rows are NOT?**

IDNBR	FNAME	LNAME	COMPNY	CITY
30368	Dan	Tuffy Jr	Tuffy Consulting	Redwing
30367	Daniel	Tuffy Sr	Tuffy Consulting	Redwing
03841	Jeff	Goeble	Kansas Farm Bureau	Kansas City
34057	Jeff	Goeble	Kansas Farm Bureau Services Inc	Kansas City

- **Understand the term *Potential Duplicate Records?***
- **Which of the two duplicate rows should be deleted?**
- **Why only four rows?**
  - Variations in representation of Company and City

Duplicate Row Case Study...

Refining the Inner Join

- **Exclude COMPNY or CITY from join criteria**
- **Exclude blank FNAME and blank LNAME**
- **Shorten columns for readability**
  - FNAME
  - LNAME
  - COMPNY
  - CITY

Duplicate Row Case Study...

Exclude CITY from Join Criteria

```
SELECT  a.idnbr,
  SUBSTR(a.fname,1,10)  AS  fname,
  SUBSTR(a.lname,1,10)  AS  lname,
  SUBSTR(a.compny,1,30)  AS  compny
       FROM  maillist  a  INNER JOIN  maillist  b
         ON  a.state  =  b.state
            AND  SUBSTR(a.fname,1,3)  =  SUBSTR(b.fname,1,3)
            AND  a.fname  >  ' '
            AND  SUBSTR(a.lname,1,3)  =  SUBSTR(b.lname,1,3)
            AND  a.lname  >  ' '
            AND  SUBSTR(a.compny,1,3)  =  SUBSTR(b.compny,1,3)
            AND  a.idnbr  < >  b.idnbr
       ORDER BY  lname,  compny,  fname
```

- **366 potential duplicate rows returned**

Duplicate Row Case Study...

Exclude COMPNY from Join Criteria

```
SELECT  a.idnbr,
  SUBSTR(a.fname,1,10)  AS  fname,
  SUBSTR(a.lname,1,10)  AS  lname,
  SUBSTR(a.city,1,15)  AS  city
       FROM  maillist  a  INNER JOIN  maillist  b
         ON  a.state  =  b.state
            AND  SUBSTR(a.fname,1,3)  =  SUBSTR(b.fname,1,3)
            AND  a.fname  >  ' '
            AND  SUBSTR(a.lname,1,3)  =  SUBSTR(b.lname,1,3)
            AND  a.lname  >  ' '
            AND  SUBSTR(a.city,1,3)  =  SUBSTR(b.city,1,3)
            AND  a.idnbr  < >  b.idnbr
       ORDER BY  lname,  compny,  fname
```

- **100 potential duplicate rows returned**

Duplicate Row Case Study...

Considerations

- **When result sets for previous two examples (exclude CITY and exclude COMPNY) were compared...**
  - Only 4 rows were in both result sets
  - Therefore multiple analysis of data with different criteria is a good idea
- **Two different tables can be compared/joined for analysis**
- **Technique will work for other than name & address tables**

Search by Sound with SOUNDEX

What is SOUNDEX?

SOUNDS LIKE Function Introduced in V4R5

- **SOUNDEX is useful for finding character strings for which the sound is known but the precise spelling is not.**
  - IBM publication: ***DB2 UDB for iSeries SQL Reference***
- ***Makes assumptions about the way that letters and combinations of letters sound that can help to search out words with similar sounds***
- **Exact spelling or case not required**
- **More flexibility than with LIKE**

What is SOUNDEX...

SOUNDEX Syntax

SOUNDEX(string-expression)

```
SELECT  lastname  FROM  address_table
   WHERE  SOUNDEX(lastname)  =  SOUNDEX('markasonnee')
   ORDER BY   lastname
```

- **Can be used anyplace in an SQL statement where a function is allowed**

Working with SOUNDEX

Searching for Marchesani

- **Phonetic spellings**
  - mark-a-son-nee
  - mark-us-annie
  - march-ez-annie

```
SELECT  lastname  FROM  address_table
   WHERE  SOUNDEX(lastname)  =  SOUNDEX('markasonnee')
   ORDER BY   lastname
```

There were 25,000 rows in address_table for these tests

Working with SOUNDEX...

Searching for Marchesani

- **Phonetic spellings**
  - mark-a-son-nee mark-us-annie march-ez-annie

```
SELECT lastname FROM address_table
   WHERE SOUNDEX(lastname) = SOUNDEX('markasonnee')
   ORDER BY lastname
```

- **23 Rows returned in result set**

Marages	Marcoccia	Marquis
Maraschky	Marcoux	Marsico
Marchesani	Marcus	Marzouk
Marchesani	Marcus	Mergy
Marchesani	Markowich	Mierzejewski
Marchesani	Marquez	Mrazek
Marchese	Marquez	Mroczek
Marcocci	Marquis	

Working with SOUNDEX...

Searching for Marchesani

- **Other phonetic spellings - Take #1**
  - mark-us
    - Same 23 rows returned
    - Is this a joke - I'm starting to get suspicious
- **Other phonetic spellings - Take #2**
  - mark
  - marks
    - 115 different rows returned, none of which are Marchesani

Working with SOUNDEX...

Testing More Complex Names

- **Other names with phonetic spellings**
  - Zborovskiy: za-bo-rov-ski, za-brov-ski
  - Zuehlsdorf: zuls-dorf
  - Barszczak: bar-sak
  - Abaunza: ab-an-sa
- **Result sets returned**
  - Zborovskiy: single row exact march
  - Zuehlsdorf: single row exact match
  - Barsczzak: 26 rows
  - Abaunza: 3 rows
- **How many rows will be returned?**

```
SELECT  COUNT(*)  FROM  address_table
   WHERE  SOUNDEX(lastname)  =  SOUNDEX('markus')
```


How Does SOUNDEX Work?

SOUNDEX Uses an Algorithm

- **Generates a four character code in format ANNN**
  - A = first (alpha) character in string expression
  - NNN = 3 digit number generated by the algorithm
- **To view 4 character SOUNDEX code**

```
SELECT  SOUNDEX(character-string),  COUNT(*)
   FROM  any-table-name
```

- **SOUNDEX code = M622**
  - markasonnee, markusannie, marchezannie, marcus
- **SOUNDEX code = M620**
  - mark, marks

How Does SOUNDEX Work...

SOUNDEX Considerations

- **First letter of phonetic spelling must be correct**
  - Will differentiate between **S** & **Z**, and **F** & **PH**
- **Sensitivity to the number of syllables**
  - If target string is multiple syllables phonetic spelling should also be multiple syllables
  - Best if target and phonetic spelling have equal number of syllables
  - Single syllable phonetic spelling can return multiple syllable result

How Does SOUNDEX Work...

SOUNDEX Performance Comparison with LIKE

- **Environment**
  - 25,000 rows in address_table
  - Lightly loaded Model 820 with 3 gigs of memory
  - Lots of disk arms
  - Interactive SQL (STRSQL command)
  - Response time measure with digital stop watch
  - Ran several iterations of each test
- **SOUNDEX**
  - marcus to find Marchesani - 2.2 seconds
- **LIKE**
  - Marc% to find Marchesani - less than 0.5 seconds

Summary

The Harder Stuff

- **Working with Edit Characters**
- **Subselect**
  - Basic Subselect
  - Scalar Subselect
  - Subselect and INSERT, UPDATE, & DELETE
  - Subselect and Derived Table
  - Subselect and CREATE TABLE
- **Summarizing Data with SELECT**
- **Identifying Potential Duplicate Rows**
- **Searching by Sound with SOUNDEX**

Summary...

The Harder Stuff...

- **From this presentation you should have a better understanding of SQL as a programming language.**
- **The more you work and play with it, the more you realize the power of SQL and what it can do for you as an application development or database manipulation tool.**
- **With a little thought and creativity you will find you can use SQL for things that at first glance you did not think possible.**

V5R2 SQL Information Sources

- **iSeries Information Center Publications - Web or CD**
  - SQL Reference
  - SQL Programming Concepts
  - SQL Programming with Host Languages
  - Query Manager Use
  - SQL Messages and Codes
- **To access Info Center on the Web**
  - **http://publib.boulder.ibm.com/iseries/v5r2/ic2924/index.htm**
    - **In left scroll bar**
      - **Click on Database - 3rd line from top**
      - **Click on Manuals**
      - **Use right scroll bar to scroll down to above SQL publication**
- **DB2 UDB for iSeries on the Web**
  - **http://www.ibm.com/servers/eserver/iseries/db2/**
- **iSeries 400 Experts Journal**
  - **http://www.iseries400experts.com**