Data Retrieval in SQL

Introduction to SQL

- Relational DBMSs do NOT always behave exactly as the relational database theory specifies
- Only relational constructs are visible to users at the external level
- But internally things may be different as the data is stored in the most optimal format possible

Introduction to SQL

- Can be used in two ways
  - interactive
  - embedded in a compiled program
- Non-procedural language
  - tell the DBMS what to get NOT how to get it

Introduction to SQL

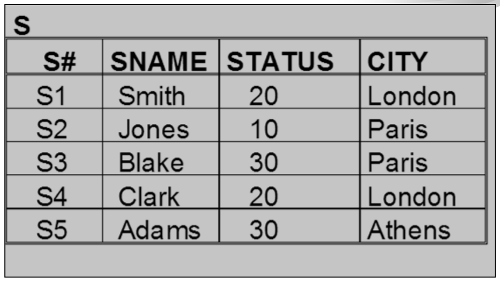
- Base table
  - a named table that really exists
  - each row is something that is stored physically
- View
  - a named table that does not have any real existence
  - derived from one or more underlying base tables

SQL General Syntax

- General syntax of SQL retrieval is

SELECT [DISTINCT | ALL] [* | [*list-of-attributes*]
FROM *list-of-tables*
[WHERE *condition*]
[GROUP BY *column-list* HAVING *condition*]
[ORDER BY *column-list*] ;

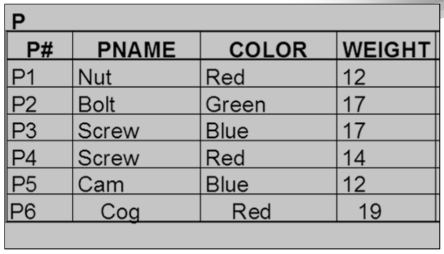
Relations to Use in SQL Examples



S

S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

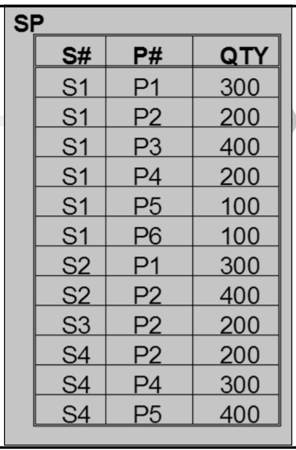
Relations to Use in SQL Examples



P

P#	PNAME	COLOR	WEIGHT
P1	Nut	Red	12
P2	Bolt	Green	17
P3	Screw	Blue	17
P4	Screw	Red	14
P5	Cam	Blue	12
P6	Cog	Red	19

Relations to Use in SQL Examples



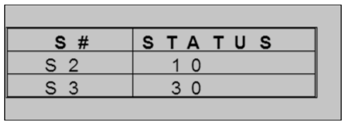
SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

SQL Examples

- EX0: Get supplier numbers and status for suppliers in Paris.

```
SELECT S#, STATUS
        FROM S
        WHERE CITY = 'Paris';
```

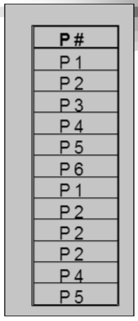


S #	S T A T U S
S 2	1 0
S 3	3 0

SQL Examples

- Ex1: Get part numbers for all parts supplied.

```
SELECT P#
   FROM SP;
```

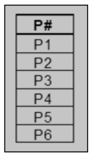


P #
P 1
P 2
P 3
P 4
P 5
P 6
P 1
P 2
P 2
P 2
P 4
P 5

SQL Examples

- Ex1a: Get part numbers for all parts supplied.

```
SELECT DISTINCT P#
   FROM SP;
```

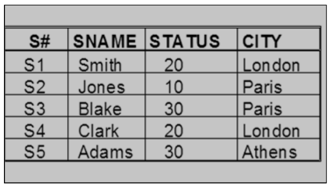


P#
P1
P2
P3
P4
P5
P6

SQL Examples

- Ex2: List the full details of all suppliers.

```
SELECT *
   FROM S;
```

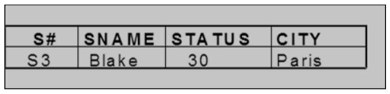


S#	SNAME	STATUS	CITY
S1	Smith	20	London
S2	Jones	10	Paris
S3	Blake	30	Paris
S4	Clark	20	London
S5	Adams	30	Athens

SQL Examples

- Ex3: List supplier numbers for all suppliers in Paris with a status greater than 20.

```
SELECT *
   FROM S
  WHERE CITY = “Paris” AND STATUS > 20;
```



S#	SNAME	STATUS	CITY
S3	Blake	30	Paris

13

SQL Examples

- Operators allowed for relational comparison:

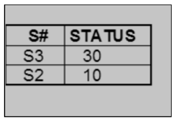
– Equal to	=
– Less than	<
– Less than or equal to	<=
– Greater than	>
– Greater than or equal to	>=
– Not equal to	<>

14

SQL Examples

- Ex4: List the supplier numbers and status for suppliers in Paris in descending order of status.

```
SELECT S#, STATUS
   FROM S
   WHERE CITY = ‘Paris’
   ORDER BY STATUS DESC;
```



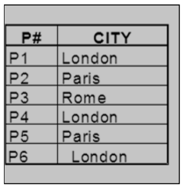
S#	STATUS
S3	30
S2	10

15

SQL Examples

- Traditional Method
  - Use the WHERE clause to define
- Ex5: For each part supplied, get the part number and names of all the cities supplying the part.

```
SELECT DISTINCT P#, CITY
        FROM SP, S
        WHERE SP.S# = S.S#;
```



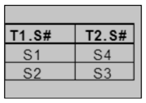
P#	CITY
P1	London
P2	Paris
P3	Rome
P4	London
P5	Paris
P6	London

16

SQL Examples

- Ex6: List the supplier numbers for all pairs of suppliers such that two suppliers are located in the same city.

```
SELECT T1.S#, T2.S#
        FROM S T1, S T2
        WHERE T1.CITY = T2.CITY
           AND T1.S# < T2.S#;
```



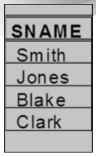
T1.S#	T2.S#
S1	S4
S2	S3

17

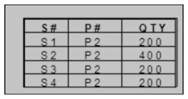
SQL Examples

- Ex7: List the supplier names for suppliers who supply part P2.

```
SELECT DISTINCT SNAME
        FROM S, SP
        WHERE S.S# = SP.S# AND
        SP.P# = ‘P2’;
```



SNAME
Smith
Jones
Blake
Clark



S#	P#	QTY
S1	P2	200
S2	P2	400
S3	P2	200
S4	P2	200

18

SQL Examples

- Ex8: List the supplier names for suppliers who supply at least one red part.

```
SELECT SNAME
        FROM S, SP, P
        WHERE S.S# = SP.S# AND
                SP.P# = P.P# AND
                P.COLOR = 'RED';
```

SNAME
Smith
Jones
Clark

19

Multiple-Row Subqueries

- Multiple-row subqueries are nested queries that can return more than one row of results to the parent query
  - Most commonly used in WHERE and HAVING clause
- Main rule
  - MUST use multiple-row operators

20

Multiple-Row Subqueries: IN Operator

- IN is a set operator used to test membership.
- The condition 'S1' **IN** ('S2', 'S3', 'S1') is true, whereas the condition 'P1' **IN** ('P2', 'P3') is false.

21

Multiple-Row Subqueries: IN Operator

- Ex7 revisited: List the supplier names for suppliers who supply part P2.

```
SELECT SNAME
        FROM S
        WHERE S# IN
                (SELECT S#
                        FROM SP
                        WHERE P# = 'P2');
```

SNAME
Smith
Jones
Blake
Clark

S#
S1
S2
S3
S4

22

Multiple-Row Subqueries: IN Operator

- The system evaluates the nested query by evaluating the nested subquery first.
  In the above query, the subquery

```
SELECT SNAME
   FROM S
   WHERE S# IN
        (SELECT S#
                FROM SP
                WHERE P# = 'P2');
```

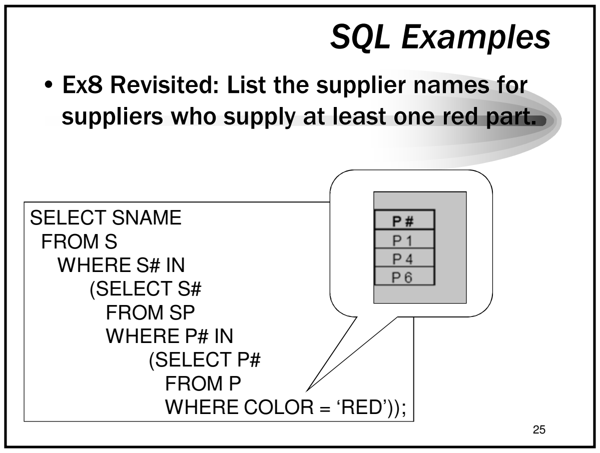
yields ('S1', 'S2', 'S3', 'S4')

23

Multiple-Row Subqueries: IN Operator

- Thus the original query becomes
  - SELECT SNAME
    FROM S
    WHERE S# IN ('S1', 'S2', 'S3', 'S4')

24



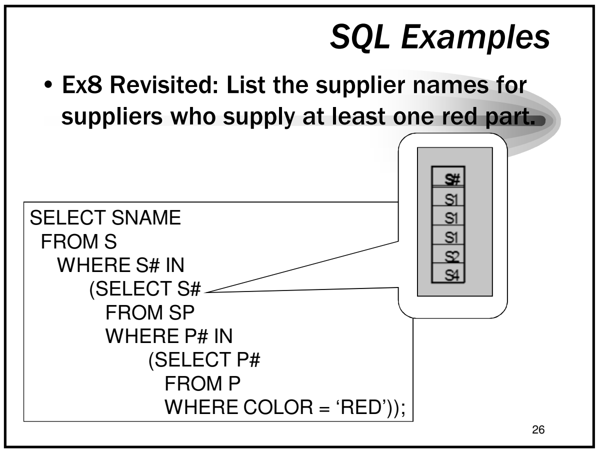
SQL Examples

- Ex8 Revisited: List the supplier names for suppliers who supply at least one red part.

```
SELECT SNAME
 FROM S
   WHERE S# IN
      (SELECT S#
         FROM SP
         WHERE P# IN
            (SELECT P#
               FROM P
               WHERE COLOR = ‘RED’));
```

P #
P 1
P 4
P 6

25



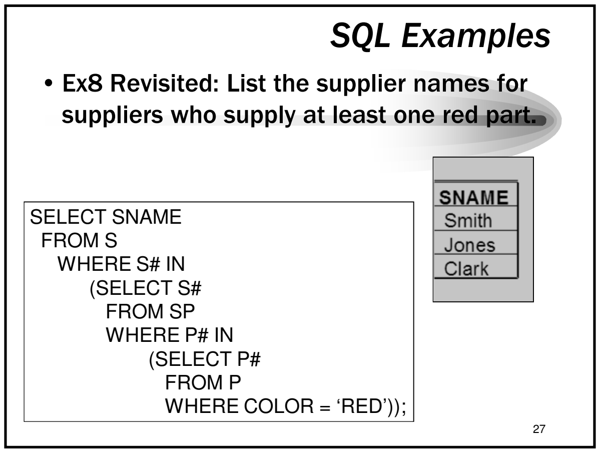
SQL Examples

- Ex8 Revisited: List the supplier names for suppliers who supply at least one red part.

```
SELECT SNAME
 FROM S
   WHERE S# IN
      (SELECT S#
         FROM SP
         WHERE P# IN
            (SELECT P#
               FROM P
               WHERE COLOR = ‘RED’));
```

S#
S1
S1
S1
S2
S4

26



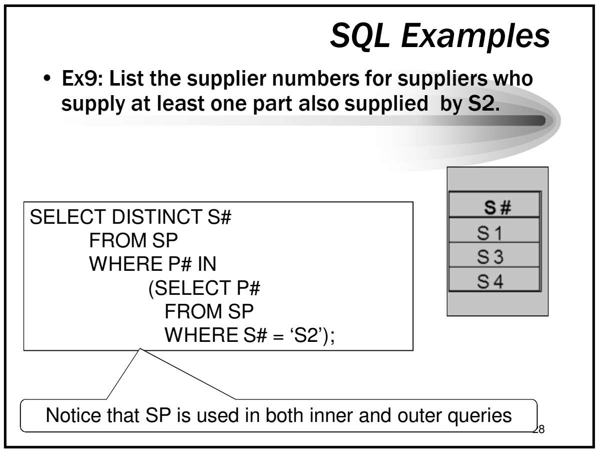
SQL Examples

- Ex8 Revisited: List the supplier names for suppliers who supply at least one red part.

```
SELECT SNAME
 FROM S
   WHERE S# IN
      (SELECT S#
         FROM SP
         WHERE P# IN
            (SELECT P#
               FROM P
               WHERE COLOR = ‘RED’));
```

SNAME
Smith
Jones
Clark

27



SQL Examples

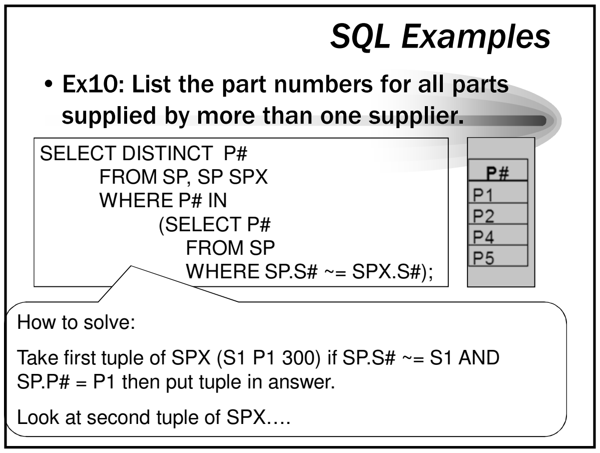
- Ex9: List the supplier numbers for suppliers who supply at least one part also supplied by S2.

```
SELECT DISTINCT S#
        FROM SP
        WHERE P# IN
                (SELECT P#
                  FROM SP
                  WHERE S# = ‘S2’);
```

S #
S 1
S 3
S 4

Notice that SP is used in both inner and outer queries

28



SQL Examples

- Ex10: List the part numbers for all parts supplied by more than one supplier.

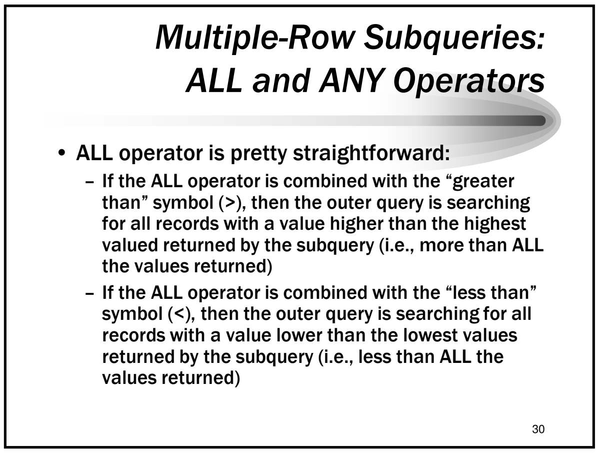
```
SELECT DISTINCT  P#
        FROM SP, SP SPX
        WHERE P# IN
                (SELECT P#
                    FROM SP
                    WHERE SP.S# ~= SPX.S#);
```

P#
P1
P2
P4
P5

How to solve:

Take first tuple of SPX (S1 P1 300) if SP.S# ~= S1 AND SP.P# = P1 then put tuple in answer.

Look at second tuple of SPX….



Multiple-Row Subqueries: ALL and ANY Operators

- ALL operator is pretty straightforward:
  - If the ALL operator is combined with the “greater than” symbol (>), then the outer query is searching for all records with a value higher than the highest valued returned by the subquery (i.e., more than ALL the values returned)
  - If the ALL operator is combined with the “less than” symbol (<), then the outer query is searching for all records with a value lower than the lowest values returned by the subquery (i.e., less than ALL the values returned)

30

Multiple-Row Subqueries: ALL and ANY Operators

- List the supplier number and quantity for suppliers who supply more quantity than any other supplier who supplies P4.

```
SELECT S#, QTY
   FROM SP
   WHERE QTY >ALL
                    (SELECT QTY
                      FROM SP
                      WHERE P# = 'P4');
```

31

Multiple-Row Subqueries: ALL and ANY Operators

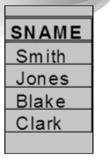
- **The <ANY operator is used to find records that have a value less than the highest value returned by the subquery**
- **The >ANY operator is used to return records that have a value greater than the lowest value returned by the subquery**
- **The =ANY operator works the same way as the IN operator does**

32

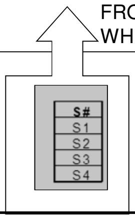
SQL Examples

- **Ex7 Revisited: List the supplier names for suppliers who supply part P2.**

```
SELECT SNAME
        FROM S
        WHERE S# = ANY
                (SELECT S#
                        FROM SP
                        WHERE P# = 'P2');
```



SNAME
Smith
Jones
Blake
Clark



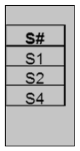
S#
S1
S2
S3
S4

33

SQL Examples

- **Ex11: List the supplier numbers for suppliers with status less than the current maximum status value in the S table.**

```
SELECT S#
        FROM S
        WHERE STATUS < ANY
                (SELECT STATUS
                        FROM S);
```



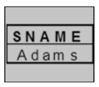
S#
S1
S2
S4

34

SQL Examples

- **Ex12: List all the names of suppliers who do not supply part P2.**

```
SELECT DISTINCT SNAME
        FROM S
        WHERE 'P2' ~= ALL
                (SELECT P#
                    FROM SP
                    WHERE SP.S# = S.S#);
```



SNAME
Adams

= ANY is equivalent to IN

~ =ALL is equivalent to NOT IN

35

Multiple-Row Subqueries: EXISTS Operator

- **The EXISTS operator is used to determine whether a condition is present in a subquery**
- **The results are boolean**
  - **TRUE if the condition exists**
  - **FALSE if it does not**

36

Multiple-Row Subqueries: EXISTS Operator

- "EXISTS (SELECT ... FROM ...)" evaluates to true
  - if and only if the result of evaluating the "SELECT ... FROM ..." is not empty.

37

SQL Examples

- Ex7 Revisited: List the supplier names for suppliers who supply part P2

```
SELECT SNAME
    FROM S
    WHERE EXISTS
         (SELECT *
             FROM SP
             WHERE SP.S# = S.S# AND
                         P# = ‘P2’);
```

SNAME
Smith
Jones
Blake
Clark

38

Multiple-Row Subqueries: NOT EXISTS Operator

- EXISTS, used in conjunction with NOT, which allows people to express two types of queries:
  - Query that involves the SET DIFFERENCE
  - Query that involves the concept of "EVERY".

39

SQL Examples

- Ex12 Revisited: List all the names of suppliers who do not supply part P2.

```
SELECT SNAME
    FROM S
    WHERE NOT EXISTS
         (SELECT *
             FROM SP
             WHERE SP.S# = S.S# AND
                         P# = ‘P2’);
```

SNAME
Adams

Another way to say the query:

Select the supplier name for suppliers such that there does not exist a supplier/part entry relating to part P2.

40

SQL Examples

- Ex13: List the supplier names for suppliers who supply all the parts.

```
SELECT SNAME
        FROM S
        WHERE NOT EXISTS
           (SELECT *
               FROM P
               WHERE NOT EXISTS
                  (SELECT *
                      FROM SP
                      WHERE SP.S# = S.S#
                          AND SP.P# = P.P#));
```

SNAME
Smith

41

SQL Examples

- Ex14: List the supplier names for suppliers who supply all the parts.

Another way to say the query: Select supplier names for suppliers such that there does not exist a part that they do not supply

SNAME
Smith

42

SQL Examples

- Ex15: Get supplier numbers for all suppliers who supply at least all those parts supplied by S2.

```
SELECT S#
        FROM S, SP
        WHERE EXISTS
                (SELECT *
                        FROM SP
                        WHERE SP.S#='S2');
```

S#
S1
S2

Notice that S2 is in the answer.

SQL Examples

- Ex16: Get the part numbers for all parts that either weigh more than 18 pounds or are currently supplied by supplier S2.

```
   SELECT P#
        FROM P
        WHERE WEIGHT > 18
UNION
   SELECT P#
        FROM SP
        WHERE S# = 'S2';
```

P#
P1
P2
P6

Group Functions

- Group functions are sometimes called multiple-row functions
- Have discussed some of these
  - SUM ([DISTINCT | ALL] n)
  - AVG ([DISTINCT | ALL] n)
  - COUNT(* | [DISTINCT | ALL] c)
  - MAX ([DISTINCT | ALL] c)
  - MIN ([DISTINCT | ALL] c)
  - STDDEV ([DISTINCT | ALL] n)
  - VARIANCE ([DISTINCT | ALL] n)

Group Functions: Group By

- In many cases, we want to apply the aggregate functions to subgroups of tuples in a relation
- Each subgroup of tuples consists of the set of tuples that have the same value for the grouping attribute(s).

Group Functions: Group By

- The function is applied to each subgroup independently
- SQL has a GROUP BY clause for specifying the grouping attributes, which must also appear in the SELECT clause

Group Functions: Group By

- When using the GROUP BY clause remember the following
  - If a group function is used in the SELECT clause, then any individual column listed in the SELECT clause must also be listed in the GROUP BY clause
  - Columns used to group data in the GROUP BY clause do not have to be listed in the SELECT clause. They are only included in the SELECT clause to have the groups identified in the output

Group Functions: Group By

- When using the GROUP BY clause remember the following
  - Column aliases cannot be used in the GROUP BY clause
  - Results returned from a SELECT statement that include a GROUP BY clause will present the results in ascending order of the column(s) listed in the GROUP BY clause. To present the results in a different order, use the ORDER BY clause.

49

Group Functions: Group By - Examples

- Ex17a: For each Part, get the P# and the total number of suppliers supplying the part.

```
SELECT P#, COUNT(*)
  FROM SP
 GROUP BY P#
```

P#	COUNT(*)
P1	2
P2	4
P3	1
P4	2
P5	2
P6	1

50

SQL Examples

- Ex17: For each part being supplied, get the part number and the total quantity.

```
SELECT P#, SUM(QTY)
       FROM SP
       GROUP BY P#;
```

P#	SUM(QTY)
P1	600
P2	100
P3	400
P4	500
P5	500
P6	100

51

Group Functions: Use of Having

- Sometimes we want to retrieve the values of these functions for only those groups that satisfy certain conditions.
- The HAVING clause is used for specifying a selection condition on groups (rather than on individual tuples)

52

SQL Examples

- Ex18: List the part numbers for all parts supplied by more than one supplier. (same as Ex11)

```
SELECT P#
       FROM SP
       GROUP BY P#
       HAVING COUNT(*) > 1;
```

P#
P1
P2
P4
P6

53

SQL Examples

- Ex19: Get the total number of suppliers.

```
SELECT COUNT(*)
       FROM S;
```

COUNT
5

54

SQL Examples

- Ex20: Get the total number of suppliers currently supplying parts.

```
SELECT DISTINCT COUNT(S#)
        FROM SP;
```

COUNT
4

55

SQL Examples

- Ex21: Get the number of shipments for part P2.

```
SELECT COUNT(*)
        FROM SP
        WHERE P# = ‘P2’;
```

COUNT
4

56

SQL Examples

- Ex22: Get the total quantity of part P2 being supplied.

```
SELECT SUM(QTY)
        FROM SP
        WHERE P# = ‘P2’;
```

COUNT
1000

57

Single-Row Subqueries

- **A single-row subquery is used when the results of the outer query are based on a single, unknown value**
- **A single-row subquery can return to the outer query only ONE row of results that consists of only ONE column**

58

Single-Row Subqueries

- **Single-row subquery in a WHERE clause**

```
SELECT Title, Cost
   FROM Books
   WHERE Cost >
                    (SELECT Cost
                       FROM Books
                       WHERE Title = ‘DATABASES’)
   AND Category = ‘COMPUTER’;
```

59

Single-Row Subqueries: Use with HAVING clause

```
SELECT Category, AVG (Retail-Cost)
   “Average Profit”
    FROM Books
    GROUP BY Category
    HAVING AVG (Retail-Cost) >
                   (SELECT AVG(Retail-Cost)
                     FROM Books
                     WHERE Category = ‘LIT’);
```

60

SQL Examples

- Ex23: List the supplier numbers for suppliers who are located in the same city as supplier S1.

```
SELECT S#
          FROM S
          WHERE CITY =
                    (SELECT CITY
                        FROM S
                        WHERE S# = ‘S1’);
```

S #
S 1
S 4

61

SQL Examples

- Ex24: Get supplier numbers for suppliers whose status is less than the current maximum status

```
SELECT S#
          FROM S
          WHERE STATUS <
                    (SELECT MAX(STATUS)
                         FROM S);
```

S#
S1
S2
S4

62

Single-Row Subqueries: In a SELECT Clause

```
SELECT Title, Retail,
               (SELECT AVG(Retail)
                  FROM Books) “Overall Average”
   FROM Books;
```

63

Retrieval using LIKE – string matching

- List suppliers whose name starts with letter S.
  - SELECT *
    FROM S
    WHERE SNAME LIKE 'S%'

64

Retrieval using LIKE – string matching

- In general, a "LIKE condition" takes the form
  - column LIKE string-literal
- Where "column" must designate a column of string type. For a given record, the condition evaluates to true if the value within the designated column conforms to the pattern specified by "literal"

65

Retrieval using LIKE – string matching

- Characters within "literal" are interpreted as follows:
  - The "-" character stands for any single character.
  - The "%" character stands for any sequence of n characters
    - (where n may be zero).
    - All other characters simply stand for themselves.

66

Retrieval using LIKE – string matching

- CITY LIKE "%BERKELEY%'
  - will evaluate to true if ADDRESS contains the string "BERKELEY" anywhere inside it.
- SNAME LIKE 'S__'
  - will evaluate to true if SNAME is exactly three character long and the first is an "S".

67

Retrieval using LIKE – string matching

- PNAME LIKE '%c___'
  - will evaluate to true if PNAME is four character long or more and the last but three is a "c"
- CITY NOT LIKE "%E%'
  - will evaluate to true if CITY does not contain an "E"

68

Retrieval using LIKE – string matching

- Using escape character
  - sname like '%\%%' will match ?
    - Abc%def
    - Abcdef
    - %
    - %%
    - %%%

69

Some Single-Row Functions

- Case Conversion Functions
  - Temporarily alters the case of data stored in a field or character string
  - Does not affect how data are stored only how data are viewed by Oracle9i during execution of a specific query
  - LOWER, UPPER and INITCAP

70

Some Single-Row Functions

- Case Conversion Functions

```
SELECT Firstname, Lastname
   FROM Customers
   WHERE LOWER(Lastname) = 'nelson';

SELECT LOWER(Firstname),
  LOWER(Lastname)
  FROM Customers
  WHERE LOWER(Lastname) = 'nelson';
```

71

Some Single-Row Functions

- Case Conversion Functions

```
SELECT Firstname, Lastname
  FROM Customers
  WHERE Lastname = UPPER ('nelson');

SELECT INITCAP(Firstname), INITCAP(Lastname)
  FROM Customers
  WHERE Lastname = 'NELSON';
```

– Converts to mixed case

72

Single-Row Functions: Character Manipulation Functions

- Determine length, extract portions of a string, or reposition a string
  - SUBSTR (c, p, l)
  - LENGTH (c)
  - LPAD (c, l, s) and RPAD (c, l, s)
  - LTRIM (c, s) and RTRIM (c, s)
  - REPLACE (c, s, r)
  - CONCAT (c1, c2)

73

Single-Row Functions: Number Functions

- Manipulates numeric data
  - Most related to trigonometry like COS, SIN, etc.
  - ROUND (n, p)
  - TRUNC (n, p)

74

Single-Row Functions: Date Functions

- Date function displays date values in a dd-mon-yy format
  - (i.e., 02-FEB-04)
- MONTHS_BETWEEN (d1, d2)
- ADD_MONTHS (d, m)
- NEXT_DAY (d, day)
- TO_DATE (d, f)

75

Single-Row Functions: Miscellaneous Functions

- NVL (x, y)
  - Where y represents the value to be substituted for if x is NULL

```
SELECT Order#, OrderDate,
   NVL(Shipdate, ’07-APR-03’),
   NVL (Shipdate, ’07-APR-03’) – OrderDate “Delay”
   FROM Orders
   WHERE Order# = 1018;
```

76

Single-Row Functions: Miscellaneous Functions

- TO_CHAR (n, ‘f’) where n is the date or number to be formatted and f is the format model to be used

```
SELECT Title,
   TO_CHAR(PubDate, ‘MONTH DD YYYY”)
     “Publication Date”,
   TO_CHAR(retail, ‘$999.99’) “Retail Price”
   FROM books
   WHERE ISBN = 0401140733;
```

77

Single-Row Functions: Miscellaneous Functions

- DECODE (V, L1, R1, L2, R2,…., D)
  - Where V is the value being searched for
    - L1 represents the first value in the list
    - R1 represents the results being returned if L1 and V are equivalent, etc., and
    - D is the default result to return if no match is found
  - Similar to CASE or IF….Then ….ELSE in many languages

```
SELECT Customer#, State,
   DECODE(State, ‘CA’, .08, ‘FL’, .07, 0) “Sales Tax Rate”
   FROM Customers;
```

78

Single-Row Functions: Miscellaneous Functions

- SOUNDEX (c)
  - Where c is the character string being referenced for phonetic representation shown as a letter and number sequence

```
SELECT Lastname, SOUNDEX (Lastname)
   FROM Customers
   WHERE Lastname LIKE ‘M%’
   ORDER BY SOUNDEX(Lastname);
```

79

Single-Row Functions: Nesting Functions

- Any of the Single-Row functions can be nested inside other Single-Row functions as long as the rules as followed
  - All arguments required for each function must be provided
  - For every open parenthesis, there must be a corresponding closed parenthesis
  - The nested, or inner, function is solved first. The result of the inner function is passed to the outer function, and the outer function is executed

80

Self-Joins

- Traditional Method
  - List table name twice using an alias for each

```
SELECT r.Firstname, r.Lastname,
      c.Lastname, c.Referred
   FROM Customers c, Customers r,
   WHERE c.Referred = r.Customer#;
```

Self-Joins

- JOIN Method
  - List table name twice using an alias for each

```
SELECT r.Firstname, r.Lastname,
c.Lastname, c.Referred
   FROM Customers c JOIN Customers r,
   ON c.Referred = r.Customer#;
```

More Joins

- Inner Join
  - This is what we have been doing with our joins
    - A join that compares the tables in the FROM clause and lists as output only those rows that satisfy the condition in the WHERE clause
- Outer Join
- Product

Outer Joins

- Keyword OUTER JOIN in SQL includes records of a table in the output even if there is no matching record in the other table
- In a sense SQL will join the “dangling” record to a NULL record in the other table

Outer Joins

- Traditional Method
  - Use the outer join operator which is a plus sign in parenthesis (+) placed in the WHERE clause immediately after the column name of the NULL tuple

```
SELECT Lastname, Firstname, Order#
   FROM Customers c, Orders o
   WHERE c.Customer# = o.Customer#(+)
   ORDER BY c.Customer#;
```

Outer Joins

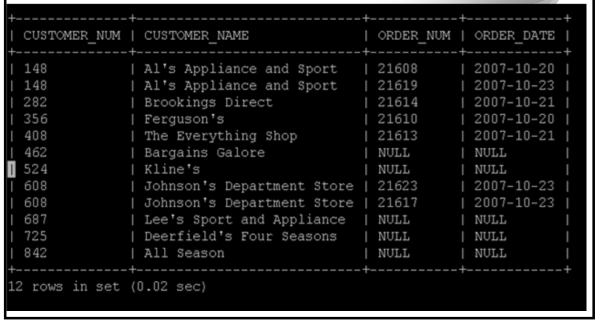
- JOIN Method
  - Include the keyword LEFT, RIGHT, or FULL with the JOIN

Outer Joins

- Display the customer number, name, order number, and order date for all orders. Include all customers in the results.

```
SELECT c.Customer_num, c.Customer_Name,
   Order_num, Order_date
   FROM Customer c
   LEFT OUTER JOIN Orders o
   on c.Customer_num = o.Customer_num
   ORDER BY c.Customer_num;
```

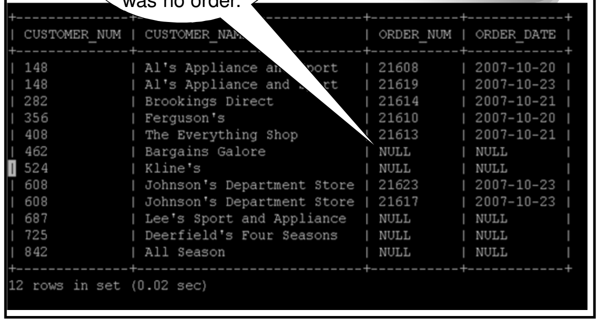
Outer Joins



```
+--------------+---------------------------+-----------+------------+
| CUSTOMER_NUM | CUSTOMER_NAME             | ORDER_NUM | ORDER_DATE |
+--------------+---------------------------+-----------+------------+
| 148          | Al's Appliance and Sport  | 21608     | 2007-10-20 |
| 148          | Al's Appliance and Sport  | 21619     | 2007-10-23 |
| 282          | Brookings Direct          | 21614     | 2007-10-21 |
| 356          | Ferguson's                | 21610     | 2007-10-20 |
| 408          | The Everything Shop       | 21613     | 2007-10-21 |
| 462          | Bargains Galore           | NULL      | NULL       |
| 524          | Kline's                   | NULL      | NULL       |
| 608          | Johnson's Department Store| 21623     | 2007-10-23 |
| 608          | Johnson's Department Store| 21617     | 2007-10-23 |
| 687          | Lee's Sport and Appliance | NULL      | NULL       |
| 725          | Deerfield's Four Seasons  | NULL      | NULL       |
| 842          | All Season                | NULL      | NULL       |
+--------------+---------------------------+-----------+------------+
12 rows in set (0.02 sec)
```

Outer Joins

Notice the NULL values where there was no order.



```
+--------------+-----------          -----+-----------+------------+
| CUSTOMER_NUM | CUSTOMER_NAM              | ORDER_NUM | ORDER_DATE |
+--------------+-------------             -+-----------+------------+
| 148          | Al's Appliance an   port  | 21608     | 2007-10-20 |
| 148          | Al's Appliance and    rt  | 21619     | 2007-10-23 |
| 282          | Brookings Direct          | 21614     | 2007-10-21 |
| 356          | Ferguson's                | 21610     | 2007-10-20 |
| 408          | The Everything Shop        21613     | 2007-10-21 |
| 462          | Bargains Galore           | NULL      | NULL       |
| 524          | Kline's                   | NULL      | NULL       |
| 608          | Johnson's Department Store| 21623     | 2007-10-23 |
| 608          | Johnson's Department Store| 21617     | 2007-10-23 |
| 687          | Lee's Sport and Appliance | NULL      | NULL       |
| 725          | Deerfield's Four Seasons  | NULL      | NULL       |
| 842          | All Season                | NULL      | NULL       |
+--------------+---------------------------+-----------+------------+
12 rows in set (0.02 sec)
```

Product

- Called Cartesian Product
  - Of two tables is the combination of all the rows of the first table and all the rows of the second table.

Product

- Form the product of the CUSTOMER and ORDERS tables.

```
SELECT c.Customer_Num, Customer_Name,
                Order_Num, Order_Date
   FROM Customer c, Orders;
```

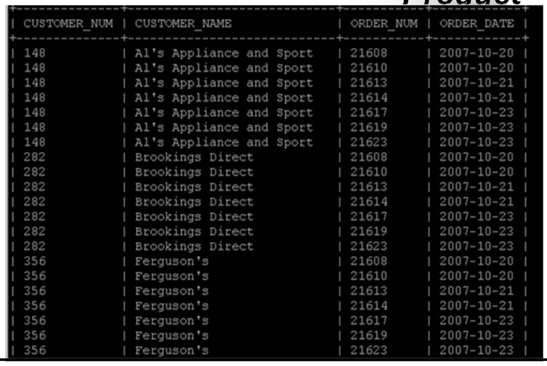
Product

- Form the product of the CUSTOMER and ORDERS tables.

```
SELECT c.Customer_Num, Customer_Name,
                Order_Num, Order_Date
   FROM Customer c, Orders;
```

Notice that there is no WHERE clause / condition.

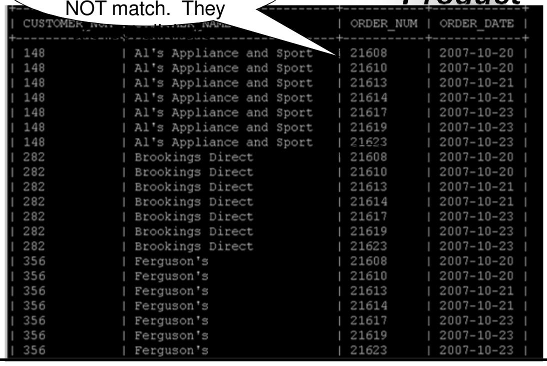
Product



```
+--------------+-----------------------------+-----------+------------+
| CUSTOMER_NUM | CUSTOMER_NAME               | ORDER_NUM | ORDER_DATE |
+--------------+-----------------------------+-----------+------------+
| 148          | Al's Appliance and Sport    | 21608     | 2007-10-20 |
| 148          | Al's Appliance and Sport    | 21610     | 2007-10-20 |
| 148          | Al's Appliance and Sport    | 21613     | 2007-10-21 |
| 148          | Al's Appliance and Sport    | 21614     | 2007-10-21 |
| 148          | Al's Appliance and Sport    | 21617     | 2007-10-23 |
| 148          | Al's Appliance and Sport    | 21619     | 2007-10-23 |
| 148          | Al's Appliance and Sport    | 21623     | 2007-10-23 |
| 282          | Brookings Direct            | 21608     | 2007-10-20 |
| 282          | Brookings Direct            | 21610     | 2007-10-20 |
| 282          | Brookings Direct            | 21613     | 2007-10-21 |
| 282          | Brookings Direct            | 21614     | 2007-10-21 |
| 282          | Brookings Direct            | 21617     | 2007-10-23 |
| 282          | Brookings Direct            | 21619     | 2007-10-23 |
| 282          | Brookings Direct            | 21623     | 2007-10-23 |
| 356          | Ferguson's                  | 21608     | 2007-10-20 |
| 356          | Ferguson's                  | 21610     | 2007-10-20 |
| 356          | Ferguson's                  | 21613     | 2007-10-21 |
| 356          | Ferguson's                  | 21614     | 2007-10-21 |
| 356          | Ferguson's                  | 21617     | 2007-10-23 |
| 356          | Ferguson's                  | 21619     | 2007-10-23 |
| 356          | Ferguson's                  | 21623     | 2007-10-23 |
```

Product

Notice that the orders and customers DO NOT match. They



```
+--------------+-----------------------------+-----------+------------+
| CUSTOMER_NUM | CUSTOMER_NAME               | ORDER_NUM | ORDER_DATE |
+--------------+-----------------------------+-----------+------------+
| 148          | Al's Appliance and Sport    | 21608     | 2007-10-20 |
| 148          | Al's Appliance and Sport    | 21610     | 2007-10-20 |
| 148          | Al's Appliance and Sport    | 21613     | 2007-10-21 |
| 148          | Al's Appliance and Sport    | 21614     | 2007-10-21 |
| 148          | Al's Appliance and Sport    | 21617     | 2007-10-23 |
| 148          | Al's Appliance and Sport    | 21619     | 2007-10-23 |
| 148          | Al's Appliance and Sport    | 21623     | 2007-10-23 |
| 282          | Brookings Direct            | 21608     | 2007-10-20 |
| 282          | Brookings Direct            | 21610     | 2007-10-20 |
| 282          | Brookings Direct            | 21613     | 2007-10-21 |
| 282          | Brookings Direct            | 21614     | 2007-10-21 |
| 282          | Brookings Direct            | 21617     | 2007-10-23 |
| 282          | Brookings Direct            | 21619     | 2007-10-23 |
| 282          | Brookings Direct            | 21623     | 2007-10-23 |
| 356          | Ferguson's                  | 21608     | 2007-10-20 |
| 356          | Ferguson's                  | 21610     | 2007-10-20 |
| 356          | Ferguson's                  | 21613     | 2007-10-21 |
| 356          | Ferguson's                  | 21614     | 2007-10-21 |
| 356          | Ferguson's                  | 21617     | 2007-10-23 |
| 356          | Ferguson's                  | 21619     | 2007-10-23 |
| 356          | Ferguson's                  | 21623     | 2007-10-23 |
```