Chapter 8

SQL SELECT Statement

REVIEW – RELATIONAL DATABASE CONCEPTS

Relational Database Concepts
Key Terms

- Table -- basic storage structure
- Column -- one kind of data in a table
- Row -- data for one table instance
- Field -- the one value found at the intersection of a row and a column
- Primary Key -- unique identifier for each row
- Foreign Key -- column that refers to a primary-key column in another table

Relational Database Concepts

- A relational database management system (RDBMS) stores and retrieves data in tables
- Tables store data in a structure of rows and columns

Accessing Data in an RDBMS

- A relational database-management system (RDBMS):
  - Organizes data into related rows and columns
  - Maintains the database
- To access data in a database:
  - The user does not need to know where the data is physically located or how to access the database
  - Structured Query Language is used to communicate with the DBMS to access the database

Properties of Tables in a Relational Database

- Each row is unique (primary key)
- Tables may be related by a common column (foreign key to primary key)
- Each table has one or more columns
- Each column has a unique name
- Values in each columns/row intersection are single-valued
- The data in each column are of the same kind (data type)
- Sequence of columns is insignificant
- Sequence of rows is insignificant

Structured Query Language (SQL)

- SQL (Structured Query Language)
  - Pronounced as separate letters, "S"–"Q"–"L", not "sequel"
  - A programming language for selecting and manipulating sets of data in a relational database
  - A nonprocedural language
    - Focus is on input/output rather than on program steps
  - Standardized by the American National Standards Institute (ANSI)
    - Unfortunately, most vendors include some proprietary SQL features into their database environment

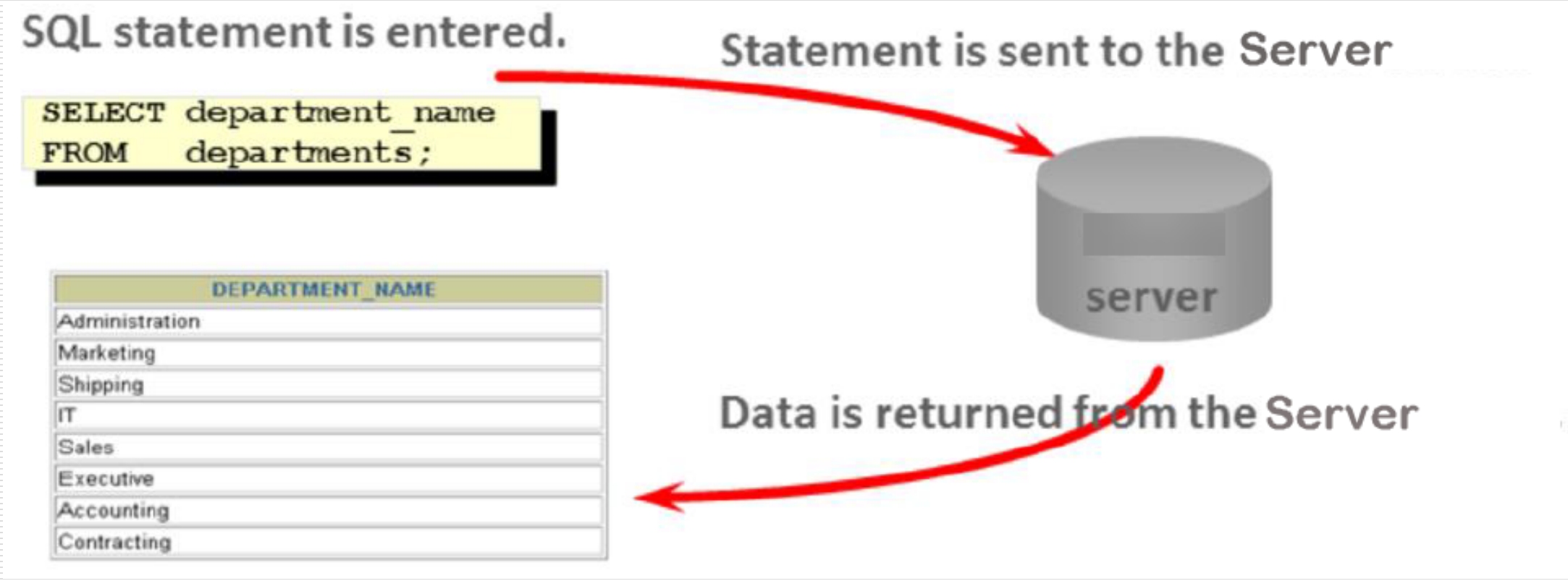
ANSI

- ANSI – American National Standards Institute
  - Structured Query Language (SQL) is the industry-standard language of relational database management systems (RDBMS)
  - Originally designed by IBM in the mid 1970s
  - Widespread use in the early 1980s
  - Became an industry standard in 1986 when it was adopted by ANSI
- Three ANSI standardizations of SQL
  - ANSI-86, ANSI-92, and ANSI-99

SELECT Statement

- The SELECT statement:
  - Retrieves data from the database
  - Allows for searching of specific data
  - Is referred to as a query

Accessing Data



Basic Format of SELECT Statement

Portion of select statement	Description
`SELECT column-names`	Specifies the columns in the SELECT statement's result set
`FROM table-list`	Specifies tables and/or views from which the result set data is returned
`WHERE search-condition`	Specifies a logical condition that must be true for a row to be included in the result set
`GROUP BY grouping-column-list`	Specifies the column(s) whose values are used to group the rows
`HAVING search-condition`	Specifies a logical condition that must be true for a group to be included in the result set
`ORDER BY order-by-column-list`	Specifies a list of columns with ascending or descending (with the DESC keyword) sequence

KEYWORD, CLAUSE, STATEMENT

Throughout this course, the following will be used:

- A keyword refers to an individual SQL command
  - For example, SELECT and FROM are keywords

```
SELECT first_name, last_name
  FROM employees;
```

KEYWORD, CLAUSE, STATEMENT

Throughout this course, the following will be used:

- A clause is a component or part of a SQL statement

```
SELECT employee_id, last_name    - is a clause
FROM employees    - is a clause
```

KEYWORD, CLAUSE, STATEMENT

Throughout this course, the following will be used:

- A statement is a combination of two or more clauses

```
SELECT first_name, last_name
    FROM employees;
```

is a SQL statement

Basic SELECT Statement

- In its simplest form, a SELECT statement must include:
  - A SELECT clause, which specifies the columns to be returned
  - A FROM clause, which specifies the table(s) containing the columns listed in the SELECT clause

```
SELECT first_name, last_name  -- SELECT Clause
  FROM employees;             -- FROM Clause
```

SELECT Statement with Column-List

- The column-list specifies the columns to be returned in the result set. The column-list can contain:
  - An asterisk (*) symbol that represents all columns in the table
  - One or more column names, specified in any order
  - Constants that are static values embedded in each row of the result set
  - SQL functions and arithmetic operators used to return a calculated value

SELECT Statement – Column List

- SELECT clause contains a column-list that identifies the column(s) to be returned in the result set

```
SELECT first_name, last_name
FROM employees;
```

```
SELECT * FROM employees;  -- All columns
```

SELECT Statement – FROM Clause

- FROM clause identifies tables and views

```
SELECT first_name, last_name
  FROM employees;
```

Selecting All Columns with SELECT *

Example 8-1 Use asterisk symbol (*) to return all columns from data set **ds81**. All rows are returned because the **WHERE** clause is not specified.

Data Set ds7_1

ID	CUSTOMER_NAME	CITY	CREDIT_LIMIT
101	Bargains Galore	Detroit	125000
102	RedHot Discount	San Diego	185500
103	NewTown Deals	Dallas	132500
104	Ajax Sales	Detroit	150000
105	BigBox Direct	Chicago	145000

SQL Statement

```
SELECT *
FROM ds7_1;
```

Result Set

ID	CUSTOMER_NAME	CITY	CREDIT_LIMIT
101	Bargains Galore	Detroit	125000
102	RedHot Discount	San Diego	185500
103	NewTown Deals	Dallas	132500
104	Ajax Sales	Detroit	150000
105	BigBox Direct	Chicago	145000

Selecting All Columns Using Column-List

Example 8-2

Specify all column names in column-list to return all columns. All rows are returned because the `WHERE` clause is not specified.

Data Set ds7_2

ID	CUSTOMER_NAME	CITY	CREDIT_LIMIT
101	Bargains Galore	Detroit	125000
102	RedHot Discount	San Diego	185500
103	NewTown Deals	Dallas	132500
104	Ajax Sales	Detroit	150000
105	BigBox Direct	Chicago	145000

SQL Statement

```
SELECT id, customer_name, city, credit_limit
FROM ds7_2;
```

Result Set

ID	CUSTOMER_NAME	CITY	CREDIT_LIMIT
101	Bargains Galore	Detroit	125000
102	RedHot Discount	San Diego	185500
103	NewTown Deals	Dallas	132500
104	Ajax Sales	Detroit	150000
105	BigBox Direct	Chicago	145000

Selecting Specific Columns Using Column-List

Example 8-3 Return specific columns for all rows

Data Set ds7_3

```
ID  CUSTOMER_NAME    CITY       CREDIT_LIMIT
--- ---------------- ---------- ------------
101 Bargains Galore  Detroit          125000
102 RedHot Discount  San Diego        185500
103 NewTown Deals    Dallas           132500
104 Ajax Sales       Detroit          150000
105 BigBox Direct    Chicago          145000
```

SQL Statement

```
SELECT customer_name, city
FROM ds7_3;
```

Result Set

```
CUSTOMER_NAME    CITY
---------------- ----------
Bargains Galore  Detroit
RedHot Discount  San Diego
NewTown Deals    Dallas
Ajax Sales       Detroit
BigBox Direct    Chicago
```

Selecting Specific Columns in a Different Order

Example 8-4 Return specific columns in a different order for all rows

Data Set ds7_4

```
ID  CUSTOMER_NAME    CITY       CREDIT_LIMIT
--- ---------------- ---------- ------------
101 Bargains Galore  Detroit          125000
102 RedHot Discount  San Diego        185500
103 NewTown Deals    Dallas           132500
104 Ajax Sales       Detroit          150000
105 BigBox Direct    Chicago          145000
```

SQL Statement

```
SELECT city, customer_name
FROM ds7_4;
```

Result Set

```
CITY       CUSTOMER_NAME
---------- ---------------
Detroit    Bargains Galore
San Diego  RedHot Discount
Dallas     NewTown Deals
Detroit    Ajax Sales
Chicago    BigBox Direct
```

Selecting One Specific Column

Example 8-5 Return one specific column for all rows

Data Set ds7_5

```
ID  CUSTOMER_NAME    CITY       CREDIT_LIMIT
--- ---------------- ---------- ------------
101 Bargains Galore  Detroit          125000
102 RedHot Discount  San Diego        185500
103 NewTown Deals    Dallas           132500
104 Ajax Sales       Detroit          150000
105 BigBox Direct    Chicago          145000
```

SQL Statement

```
SELECT customer_name
FROM ds7_5;
```

Result Set

```
CUSTOMER_NAME
---------------
Bargains Galore
RedHot Discount
NewTown Deals
Ajax Sales
BigBox Direct
```

SQL Statement Line Spacing

```
SELECT customer_name FROM customers;
```

Figure 8-6 SQL statement with no line spacing

```
SELECT customer_name
FROM customers;
```

Figure 8-7 SQL statement with line spacing

```
SELECT customer_name
    FROM customers;
```

Figure 8-8 SQL statement with indentation

Using the Correct Syntax

- Syntax is the set of rules used to create SQL statements, which includes the correct spelling and arrangement of keywords

Using the Correct Syntax
Correct Spelling on Keyword

- The following statement has a spelling error:

```
SELCT *
  FROM departments;
```

```
SQL State: 42601
Vendor Code: -104
Message: [SQL0104] Token SELCT was not valid.
```

Figure 8-9 SQL syntax error – spelling error on keyword SELECT

Using the Correct Syntax
Correct Spelling on Table & Column Names

```
SELCT *
  FROM departmets;
```

```
SQL State: 42704
Vendor Code: -204
Message: [SQL0204] DEPARTMETS in COMMONLIB type *FILE not found.
```

Figure 8-10 SQL syntax error – spelling error on table name

```
SELECT cust_name
  FROM customers;
```

```
[SQL0206] Column or global variable CUST_NAME not found.
```

Figure 8-11 SQL syntax error – spelling error on column name

Computed Columns - Arithmetic Operators

- Arithmetic operators are used to perform calculations on numeric values:

Arithmetic operator	Description
( )	Parentheses/Brackets
^	Exponent/Power Of
*	Multiplication
/	Division
+	Addition
-	Subtraction

Figure 8-12 Arithmetic operators

Order of Operations – Which is it?

- PEMDAS
- BEDMAS
- BODMAS
- BIDMAS

PEMDAS	BEDMAS	BODMAS	BIDMAS
(Parentheses **)**	**(** Brackets **)**	**(** Brackets **)**	**(** Brackets **)**
Exponents^2	**E**xponents^2	P**O**wer^2	**I**ndices^2
Multiplication **x**	**D**ivision **÷**	**D**ivision **÷**	**D**ivision **÷**
Division **÷**	**M**ultiplication **x**	**M**ultiplication **x**	**M**ultiplication **x**
Addition **+**	**A**ddition **+**	**A**ddition **+**	**A**ddition **+**
Subtraction **-**	**S**ubtraction **-**	**S**ubtraction **-**	**S**ubtraction **-**

Arithmetic Order of Operations

- The order of operations is the order in which the DBMS evaluates different operators in the same expression

() ^ * / + -

  - Multiplication and division take priority over addition and subtraction
  - Operators of the same priority (* /) (+ -) are evaluated from left to right
  - Parentheses or brackets are used to force the expression within parentheses/brackets to be evaluated first

Arithmetic Order of Operations

- What's the answer:
- $15 \times 2 / (5-3) - 2^3 + 4 \times 2$

Arithmetic Order of Operations

15

Computed Columns

- A computed column is:
  - A column specified in the column-list that does not exist in the database
  - A value that is calculated using data from existing columns in the table
- Computed columns do not create new columns in the database or change the actual data values
- The results of the calculations only appear in the result set

Using Arithmetic Operators

Example 8-6

Return customer name and a new calculated column showing a ten percent increase in the current credit limit. The new calculated column is calculated as `credit_limit * 1.10`

Data Set ds7_6

```
ID  CUSTOMER_NAME    CITY       CREDIT_LIMIT
--- ---------------- ---------- ------------
101 Bargains Galore  Detroit          125000
102 RedHot Discount  San Diego        185500
103 NewTown Deals    Dallas           132500
104 Ajax Sales       Detroit          150000
105 BigBox Direct    Chicago          145000
```

SQL Statement

```
SELECT customer_name, credit_limit * 1.10
FROM ds7_6;
```

Result Set

```
CUSTOMER_NAME    00002
---------------- ---------
Bargains Galore  137500.00
RedHot Discount  204050.00
NewTown Deals    145750.00
Ajax Sales       165000.00
BigBox Direct    159500.00
```

Using Arithmetic Operators

- Calculate the new annual salary by giving employees a $100 raise each month
- Is this correct?

```
SELECT last_name, salary, 12 * salary + 100
FROM employees;
```

Solution

Operator Precedence

```
SELECT last_name, salary,
       12*salary +100
FROM employees;
```

LAST_NAME	SALARY	12*SALARY+100
King	24000	288100
Kochhar	17000	204100
De Haan	17000	204100
Whalen	4400	52900
Higgins	12000	144100
Gietz	8300	99700

Using Parentheses

```
SELECT last_name, salary,
       12*(salary +100)
FROM employees;
```

LAST_NAME	SALARY	12*(SALARY+100)
King	24000	289200
Kochhar	17000	205200
De Haan	17000	205200
Whalen	4400	54000
Higgins	12000	145200
Gietz	8300	100800

NULL Values

- What is a NULL value?

NULL Values

- A *null* is a value that is *unavailable*, *unassigned*, or *unknown*
  - NULL is **not** the same as **zero**. Zero is a number
  - NULL is **not** a **space**. Space is a character

- Sometimes, the value for a column in a database table is not known
- Relational databases use NULL values to represent unknown values

NULL Values

- If any column value in an arithmetic expression is NULL, the result is NULL
- Dividing by NULL, results is NULL
- Dividing by zero, returns a divide by zero error

- Depending upon the DBMS, NULL values may appear as NULL, spaces, a dash (-), or other representation

NULL Values in Arithmetic Expressions

```
SELECT salary, commission_pct, salary * commission_pct
FROM employees;
```

SALARY	COMMISSION_PCT	00003
2500	(null)	(null)
10500	.2	2100
11000	.3	3300
(null)	.2	(null)
7000	.15	1050
(null)	(null)	(null)

Column Alias

- Renames a column heading in the output
  - Display a column name that is easier to understand
- Is useful with calculations
- Immediately follows the column name
- May have the optional AS keyword between the column name and alias
- Requires double quotation marks if the alias contains spaces or special characters, or is case-sensitive

Column Alias

- Without aliases, column names are:
  - Column names from the table in upper case
  - A number or name showing an arithmetic operation such as salary*commission_pct

```
SELECT salary, commission_pct, salary * commission_pct
FROM employees;
```

SALARY	COMMISSION_PCT	00003
2500	(null)	(null)
10500	.2	2100
11000	.3	3300
(null)	.2	(null)
7000	.15	1050
(null)	(null)	(null)

Column Alias Using Optional AS Clause

Example 8-7

Return customer name, credit limit, and a new calculated column showing a ten percent increase in the current credit limit. The new calculated column is calculated as **credit_limit * 1.10.** Alias names are used to describe the output in the result set.

Data Set ds7_7

```
ID  CUSTOMER_NAME    CITY       CREDIT_LIMIT
--- ---------------- ---------- ------------
101 Bargains Galore Detroit          125000
102 RedHot Discount San Diego        185500
103 NewTown Deals   Dallas           132500
104 Ajax Sales      Detroit          150000
105 BigBox Direct   Chicago          145000
```

SQL Statement

```
SELECT customer_name, credit_limit AS old_credit_limit,
       credit_limit * 1.10 AS new_credit_limit
FROM ds7_7;
```

Result Set

```
CUSTOMER_NAME    OLD_CREDIT_LIMIT NEW_CREDIT_LIMIT
---------------- ---------------- ----------------
Bargains Galore            125000        137500.00
RedHot Discount            185500        204050.00
NewTown Deals              132500        145750.00
Ajax Sales                 150000        165000.00
BigBox Direct              145000        159500.00
```

Column Alias – No AS Clause

Example 8-8 Use alias names in the result set

Data Set ds7_8

```
ID  CUSTOMER_NAME    CITY       CREDIT_LIMIT
--- ---------------- ---------- ------------
101 Bargains Galore Detroit          125000
102 RedHot Discount San Diego        185500
103 NewTown Deals   Dallas           132500
104 Ajax Sales      Detroit          150000
105 BigBox Direct   Chicago          145000
```

SQL Statement

```
SELECT customer_name AS customer,
       city AS ship_city,
       credit_limit maximum_credit
FROM ds7_8;
```

Result Set

```
CUSTOMER        SHIP_CITY MAXIMUM_CREDIT
--------------- --------- --------------
Bargains Galore Detroit           125000
RedHot Discount San Diego         185500
NewTown Deals   Dallas            132500
Ajax Sales      Detroit           150000
BigBox Direct   Chicago           145000
```

Column Alias – Using Quotes

Example 8-9

Return `customer_name`, and `available_credit` for all rows in the `CUSTOMERS` table. `Available Credit` is an alias for the result of the calculation `credit_limit - balance`.

Data Set ds7_9

```
CUSTOMER_NAME   CREDIT_LIMIT BALANCE
--------------- ------------ ---------
Bargains Galore       125000 110083.00
RedHot Discount       185500 120387.00
NewTown Deals         132500 109635.00
Ajax Sales            150000 118630.00
BigBox Direct         145000 123122.00
```

SQL Statement

```
SELECT customer_name AS "Customer",
       credit_limit - balance AS "Available Credit"
FROM ds7_9;
```

Result Set

```
Customer        Available Credit
--------------- ----------------
Bargains Galore          14917.00
RedHot Discount          65113.00
NewTown Deals            22865.00
Ajax Sales               31370.00
BigBox Direct            21878.00
```

Concatenation Operator

- Concatenation means to connect or link two items together
- Two vertical bars (||), sometimes referred to as "pipes"
- Columns on either side of the || operator are combined to make a single output column
- Can link columns to other columns, arithmetic expressions, or constant values to create a character expression
- The resulting value is a character string

Concatenation Operator

|| ' ' || is used to insert a space between text for readable output

```
SELECT department_id || ' ' || department_name
FROM departments;
```

DEPARTMENT_ID||''||DEPARTMENT_NAME
10 Administration
20 Marketing
50 Shipping
60 IT
80 Sales
90 Executive
110 Accounting
190 Contracting

Concatenation and Column Aliases

- A column alias can be used to provide a column name for a concatenation operation

```
SELECT department_id || ' ' || department_name AS "Department Info"
FROM departments;
```

Department Info
10 Administration
20 Marketing
50 Shipping
60 IT
80 Sales
90 Executive
110 Accounting
190 Contracting

Concatenation Operator

Example 8-10 Use the concatenation (||) operator to return first and last names concatenated together with an alias "**Employee Name**"

Data Set ds7_10

```
ID  FIRST_NAME LAST_NAME
--- ---------- ---------
201 Martina    Lopez
202 Avery      Brown
203 Chetan     Patel
204 Logan      Moore
205 Wang       Fang
```

SQL Statement

```
SELECT first_name || last_name AS "Full Name"
FROM ds7_10;
```

Result Set

```
Full Name
------------
MartinaLopez
AveryBrown
ChetanPatel
LoganMoore
WangFang
```

Concatenation Operator – Adding a Space

Example 8-11	Use the concatenation (||) operator to return first and last names concatenated together with an alias "**Full Name**" Insert a space between the first and last names.
Data Set ds7_11	ID FIRST_NAME LAST_NAME --- ---------- --------- 201 Martina Lopez 202 Avery Brown 203 Chetan Patel 204 Logan Moore 205 Wang Fang
SQL Statement	SELECT first_name || ' ' || last_name AS "Full Name" FROM ds7_11;
Result Set	Full Name ------------- Martina Lopez Avery Brown Chetan Patel Logan Moore Wang Fang

Concatenation and Literal Values

- Literal Values
  - A fixed data value such as a character, number, or date
  - Using concatenation and literal values, output can appear as a sentence or statement

  - Examples:
    - 'dollars'
    - 1000
    - 'January 1, 2009'

Concatenation and Literal Values

- Literal values can be included in the SELECT list with the concatenation operator
- Characters and dates must be enclosed in single quotes (')
- Number literals are not enclosed in single quotes
- Every row returned from a query with literal values will have the same character string in it
- Consider the example on next slide

Concatenation and Literal Values

- The strings, 'has a salary of' and 'dollars ', are examples of literals

Example 8-12	Use the concatenation operator and literal values to return a string

Data Set ds7_12

```
ID  FIRST_NAME LAST_NAME SALARY
--- ---------- --------- ------
201 Martina    Lopez      65000
202 Avery      Brown      52000
203 Chetan     Patel      74500
204 Logan      Moore      45700
205 Wang       Fang       59200
```

SQL Statement

```
SELECT first_name || ' ' || last_name || ' has a salary of ' ||
       salary || ' dollars' AS "Employee Salary"
FROM ds7_12;
```

Result Set

```
Employee Salary
----------------------------------------
Martina Lopez has a salary of 65000 dollars
Avery Brown has a salary of 52000 dollars
Chetan Patel has a salary of 74500 dollars
Logan Moore has a salary of 45700 dollars
Wang Fang has a salary of 59200 dollars
```

Concatenation and Literal Values

- Numbers as literal values

```
SELECT last_name ||
       ' has a ' || 1 ||
       ' year salary of ' ||
       salary*12 || ' dollars.' AS Pay
FROM employees;
```

PAY
King has a 1 year salary of 288000 dollars.
Kochhar has a 1 year salary of 204000 dollars.
De Haan has a 1 year salary of 204000 dollars.
Whalen has a 1 year salary of 52800 dollars.
Higgins has a 1 year salary of 144000 dollars.

Using DISTINCT to Eliminate Duplicate Rows

- Many times, you may want to know how many unique instances of something exist
- For example, what if the company wanted a list of cities where they ship products to customers

List of cities the company ships to

Example 8-13 — Return a list of cities (**city**) where customers are located

Data Set ds7_13

```
ID  CUSTOMER_NAME    CITY
--- ---------------- ---------
101 Bargains Galore  Detroit
102 RedHot Discount  San Diego
103 NewTown Deals    Dallas
104 Ajax Sales       Detroit
105 BigBox Direct    Chicago
106 Mainstreet Inc   Dallas
107 Riverside Mfg    Chicago
108 Cube Industries  Dallas
109 LowCost Shops    San Diego
110 Sterling Mfg     Chicago
```

SQL Statement

```
SELECT city
FROM ds7_13;
```

Result Set

```
CITY
---------
Detroit
San Diego
Dallas
Detroit
Chicago
Dallas
Chicago
Dallas
San Diego
Chicago
```

Using DISTINCT to Eliminate Duplicate Rows

- `DISTINCT` - used in the column-list to eliminate duplicate values
- As a result, there are fewer rows returned
- The keyword DISTINCT must appear directly after the SELECT keyword
- The `DISTINCT` qualifier affects all the columns listed in the column-list and returns every distinct combination of the columns in the SELECT clause

Using DISTINCT to Eliminate Duplicate Rows

Example 8-14 Use the **DISTINCT** operator to list the cities where there is at least one customer

Data Set ds7_14

```
ID  CUSTOMER_NAME    CITY
--- ---------------- ---------
101 Bargains Galore  Detroit
102 RedHot Discount  San Diego
103 NewTown Deals    Dallas
104 Ajax Sales       Detroit
105 BigBox Direct    Chicago
106 Mainstreet Inc   Dallas
107 Riverside Mfg    Chicago
108 Cube Industries  Dallas
109 LowCost Shops    San Diego
110 Sterling Mfg     Chicago
```

SQL Statement

```
SELECT DISTINCT city
FROM ds7_14;
```

Result Set

```
CITY
---------
Dallas
Chicago
Detroit
San Diego
```

Using DISTINCT to Eliminate Duplicate Rows

- Consider - Want a list of all of the departments for which there are employees?

```
SELECT department_id
FROM employees;
```

DEPARTMENT_ID
90
90
90
10
110
110
80
80
80
-

- Does this give us what we want?

Using DISTINCT to Eliminate Duplicate Rows

- DISTINCT - Want to know how many unique instances of something exist

```
SELECT DISTINCT department_id
FROM employees;
```

DEPARTMENT_ID
-
90
20
110
80
50
10
60

Using DISTINCT to Eliminate Duplicate Rows

- The DISTINCT qualifier affects all listed columns and returns every distinct combination of the columns in the SELECT clause
- The keyword DISTINCT must appear directly after the SELECT keyword

```
SELECT DISTINCT department_id
FROM employees;
```

DEPARTMENT_ID
-
90
20
110
80
50
10
60

Using DISTINCT with Multiple Columns

Example 8-16 Return a list of reps and their customers for which there is at least one order

Data Set ds7_16

```
REP_ID|CUSTOMER_NAME  |CITY     |ORDER_TOTAL|
------|---------------|---------|-----------|
   101|Bargains Galore|Detroit  |     125000|
   101|Bargains Galore|Detroit  |     134700|
   102|RedHot Discount|San Diego|     185500|
   101|Treetop Inc    |Toronto  |     182500|
   102|RedHot Discount|San Diego|     185500|
   103|NewTown Deals  |Dallas   |     132500|
   103|NewTown Deals  |Dallas   |     112500|
   101|Treetop Inc    |Toronto  |     168000|
```

SQL Statement

```
SELECT DISTINCT rep_id, customer_name
FROM ds7_16;
```

Result Set

```
REP_ID CUSTOMER_NAME
------ --------------
   103 NewTown Deals
   101 Treetop Inc
   101 Bargains Galore
   102 RedHot Discount
```


Q & A
Questions &
Answers