

Database Design and SQL

Single Table Queries

CSAM - 2021S

Sagara Samarawickrama

Section 1 & 2

Chapter Objectives

- Use SQL commands to retrieve data from a database table
- Use literals in the selected list
- Use computed columns in the selected list.
- Use the AS clause
- Use Simple and Compound conditions in WHERE clauses
- Use the DISTINCT keyword
- Use the CONCAT,BETWEEN,LIKE and IN operations
- Sort the results of a SELECT statement using the ORDER BY clause

Single Table Queries

Basic Format of the SELECT Statement:

The SQL SELECT statement is used to query a database and return a result set containing rows from one or more tables or views.

```
SELECT column_names  
FROM table or view_name  
WHERE search_condition  
GROUP BY column_names  
HAVING search_condition  
ORDER BY column_name;
```

The SELECT and FROM keywords must be specified; the other keywords are optional.

Single Table Queries

The parts of a SELECT statement serve the following purposes:

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

Single Table Queries

Example 6-1: Return all row and column from DEPARTMENTS table.

```
SELECT *  
FROM departments;
```

Results

DEPARTMENT_CODE	DEPARTMENT_NAME	MANAGER_ID
AD	Administration	NULL
AC	Accounting	NULL
MK	Marketing	NULL

Example 6-2: Return customer_name for all rows in the CUSTOMERS table.

```
SELECT customer_name  
FROM customers;
```

Results

CUSTOMER_NAME
Smith Mfg.
Bolt Co.
Ajax Steel Inc.
Bluewater

Single Table Queries

Select Multiple Columns From a TABLE

Example 6-3: Return customer_id, customer_name, and discount for all rows in the CUSTOMERS table.

```
SELECT customer_id, customer_name, discount  
FROM customers;
```

Results

CUSTOMER_ID	CUSTOMER_NAME	DISCOUNT
133568	Smith Mfg.	0.050
246900	Bolt Co.	0.020
275978	Ajax Steel Inc.	NULL
499320	Bluewater Inc.	0.015
499921	Bell Bldg.	0.010
518980	London Inc.	0.050
663456	Alpine Inc.	

Single Table Queries

Computed Columns

A computed column is a column in the result set that does not exist in the table. Instead a computed column is calculated using a data from existing columns in the table.

Arithmetic operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division

Single Table Queries

Example 6-4: Return order_id, customer_id, order_total, and discount for all rows in the ORDERS table. Discount is a computed value calculated as order_total * .05.

```
SELECT order_id, customer_id, order_total,  
       (order_total * .05) AS discount  
FROM orders;
```

Results

ORDER_ID	CUSTOMER_ID	ORDER_TOTAL	DISCOUNT
234112	499320	35.00	1.7500
234113	888402	278.75	13.9375
234114	499320	78.90	3.9450
234115	890002		

```
SELECT customer_id, customer_name,  
       (credit_limit - balance) AS "Available Credit"  
FROM customers;
```

Results

CUSTOMER_ID	CUSTOMER_NAME	Available Credit
133568	Smith Mfg.	
246900	Bolt Co.	148002.00
275978	Ajax Steel Inc.	224896.45

Single Table Queries

Column Aliases

An alias is a way to rename column heading in the output. Aliases can be used to better describe the contents of a column in the result set.

```
SELECT first_name      AS "First Name",  
       middle_initial  Initial,  
       last_name       "Last Name"  
FROM employees;
```

Results

First Name	INITIAL	Last Name
Lauren	M	Alexander
Lisa	L	James
Dave		Bernard
Steve	L	Carr
Marg	A	

Single Table Queries

Concatenation Operator

Concatenation means to connect two items together.

SELECT first_name || last_name AS "Employee Name"
FROM employees;

Employee Name
StevenKing
NeenaKochhar
LexDe Haan

SELECT first_name || ' ' || last_name AS "Employee Name"
FROM employees;

Employee Name
Steven King
Neena Kochhar
Lex De Haan

Single Table Queries

DISTINCT keyword

When the SELECT statement does not include the primary key columns , the result set may contain duplicate rows. To eliminate duplicate rows from the result set ,the SELECT keyword is followed with the DISTINCT keyword.

Example 6-12: Return ship_city for all rows in the CUSTOMERS table. Use the DISTINCT operator to list each city only once.

```
SELECT DISTINCT ship_city  
FROM customers;
```

Results

SHIP_CITY

Chicago
Toronto
Albany
Portland

Single Table Queries

Using the WHERE Clause to limit row selection

Sometimes, a request is made to retrieve only one row or set of rows that meet a search condition

```
SELECT customer_id, customer_name  
FROM customers  
WHERE ship_city = 'Boston';
```

Results

CUSTOMER_ID	CUSTOMER_NAME
-------------	---------------

518980	London Inc.
--------	-------------

663456	Alpine Inc.
--------	-------------

Single Table Queries

The Where condition can return zero, one or more rows. The Where clause specifies which columns or rows will be returned, based on the criteria described after the WHERE keyword. A simple condition compares two values, using one of the comparison operators listed below

Comparison operator	Description
=	Equal
>	Greater than
<	Less than
>=	Greater than or equal
<=	Less than or equal
<>	Not equal

Single Table Queries

Example 6-16: Test for a NULL condition

Return customer_id, customer_name, and discount for all rows in the CUSTOMERS table where discount is unknown.

```
SELECT customer_id, customer_name, discount
FROM customers
WHERE discount IS NULL;
```

Results

CUSTOMER_ID	CUSTOMER_NAME	DISCOUNT
275978	Ajax Steel Inc.	NULL
781010	Bluewater Mfg.	NULL

Example 6-17: Test for a NOT NULL condition

Return customer_id, customer_name, and discount for all rows in the CUSTOMERS table where discount is not null.

```
SELECT customer_id, customer_name, discount
FROM customers
WHERE discount IS NOT NULL;
```

Results

CUSTOMER_ID	CUSTOMER_NAME	DISCOUNT
133568	Smith Mfg.	0.050
246900	Bolt Co.	0.020
499320	Bluewater Inc.	

Single Table Queries

Using a Computed Value with a Where Clause

A computed value can be used in WHERE clause.

Example 6-18: Using a computed column with a WHERE clause

Return customer_id, customer_name, and available_credit (credit_limit - balance) for all rows in the CUSTOMERS table where available credit is greater than 100,000.

```
SELECT customer_id AS "Customer",  
       customer_name AS "Customer Name",  
       credit_limit - balance AS "Available Credit"  
FROM customers  
WHERE (credit_limit - balance) > 100000;
```

Results

Customer	Customer Name	Available Credit
133568	Smith Mfg.	148002.00
246900	Bolt Co.	224896.45

Single Table Queries

Using Compound Conditions

The WHERE clause also may contain a compound condition that specifies two or more conditions with AND or OR connectors.

Example 6-19: Compound search condition

Return customer_name, ship_city, and discount for all rows in the CUSTOMERS table where ship_city = Albany AND discount > 0.

```
SELECT customer_name, ship_city, discount
FROM customers
WHERE ship_city = 'Albany'
AND discount > 0;
```

Results

CUSTOMER_NAME	SHIP_CITY	DISCOUNT
Seaworthy	Albany	0.010

Single Table Queries

Negating a Conditions

To negate a condition, the NOT logical operator can be specified at the beginning of any condition or before conditions connected by AND or OR.

Example 6-20: Using NOT to negate a search condition

Return customer_name, ship_city, and discount for all rows in the CUSTOMERS table where the ship_city is not equal to Albany.

```
SELECT customer_name, ship_city, discount
FROM customers
WHERE NOT ( ship_city = 'Albany');
```

Results

CUSTOMER_NAME	SHIP_CITY	DISCOUNT
Smith Mfg.	Chicago	0.050
Bolt Co.	Toronto	0.020
Bluewater Inc.	Portland	0.015

Single Table Queries

Comparison Operators

In addition to the comparison operators equal to (=), less than (<), and greater than (>), SQL has other operators that add functionality for retrieving sets of data. These include:

- BETWEEN ... AND
- IN
- LIKE

Single Table Queries

Comparison Operators

In addition to the comparison operators equal to (=), less than (<), and greater than (>), SQL has other operators that add functionality for retrieving sets of data. These include:

- BETWEEN ... AND
- IN
- LIKE

Single Table Queries

BETWEEN ...AND Operators

Example 6-22 The BETWEEN operator

Retrieve customer_id, customer_name, and discount for all rows in the CUSTOMERS table where discount is between 0.01 and 0.02.

```
SELECT customer_id, customer_name, discount
FROM customers
WHERE discount BETWEEN 0.01 AND 0.02;
```

Results

CUSTOMER_ID	CUSTOMER_NAME	DISCOUNT
246900	Bolt Co.	0.020
499320	Bluewater Inc.	0.015

Single Table Queries

NOT BETWEEN Operators

Example 6-23: The NOT BETWEEN operator

Retrieve customer_id, customer_name, and discount for all rows in the CUSTOMERS table where discount is NOT BETWEEN 0.01 and 0.02.

```
SELECT customer_id, customer_name, discount
FROM customers
WHERE discount NOT BETWEEN 0.01 AND 0.02;
```

Results

CUSTOMER_ID	CUSTOMER_NAME	DISCOUNT
133568	Smith Mfg.	0.050
518980	London Inc.	0.050

Single Table Queries

IN Operators

Example 6-24: Using the IN operator with a WHERE clause

Retrieve customer_id, customer_name, and ship_city for all rows in the CUSTOMERS table where ship_city is either Detroit, Portland, or Boston.

```
SELECT customer_id, customer_name, ship_city
FROM customers
WHERE ship_city IN ( 'Detroit', 'Portland', 'Boston' );
```

Results

CUSTOMER_ID	CUSTOMER_NAME	SHIP_CITY
499320	Bluewater Inc.	Portland
499921	Bell Bldg.	Detroit

Single Table Queries

LIKE Operator

The percent sign(%) specifies a match to a string of any length

Underscore(_) specifies a match on a single character

```
SELECT employee_id, first_name
FROM employees
WHERE first_name LIKE 'R%';
```

Results	
EMPLOYEE_ID	FIRST_NAME
116	Robert
139	Rick

```
SELECT employee_id, first_name
FROM employees
WHERE first_name LIKE '_ick%';
```

Results	
EMPLOYEE_ID	FIRST_NAME
139	Rick

Single Table Queries

Logical Operator

A logical operator combines the result of two or more conditions to produce a single result.

- ❖ AND Operator: Returns TRUE if both conditions are true
- ❖ OR Operator : Returns TRUE if either condition is true
- ❖ NOT Operator : Returns TRUE if the condition is false

Single Table Queries

AND Operator

```
SELECT first_name, last_name, department_code, salary
FROM employees
WHERE department_code = 'IT' AND salary > 23500;
```

Results

FIRST_NAME	LAST_NAME	DEPARTMENT_CODE	SALARY
Greg	Zimmerman	IT	31500.00
Dave	Bernard	IT	24000.00
Rick	Peters	IT	28750.00

Single Table Queries

OR Operator

```
SELECT first_name, last_name, department_code, salary  
FROM employees  
WHERE department_code = 'IT' OR salary > 23500;
```

Results

FIRST_NAME	LAST_NAME	DEPARTMENT_CODE	SALARY
Lauren	Alexander	TR	45000.00
Lisa	James	MA	65000.00
Dave	Bernard	HT	60000.00
Steve	Carr	VG	55000.00
Marg	Horner	MA	45000.00

Single Table Queries

NOT Operator

```
SELECT first_name, last_name, department_code, salary
FROM employees
WHERE department_code NOT IN ('MA', 'VG', 'HT');
```

Results

FIRST_NAME	LAST_NAME	DEPARTMENT_CODE	SALARY
Lauren	Alexander	TR	45000.00
Jim	Best	SA	24000.00
Greg	Zimmerman	IT	31500.00
		IT	24000.00

Conclusion

