



Performing a Conditional Search

Database Fundamentals

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Welcome to Performing a Conditional Search.

What you will learn

At the core of the lesson

You will learn how to do the following:

- Search by using one or more conditions.
- Search for a range of values and NULL values.
- Search data based on string patterns.

Key terms:

- WHERE clause
- Comparison operator
- Arithmetic operator
- Logical operator
- Wildcard
- Column alias
- NULL value



In this module, you will learn how to do the following:

- Search using one or more conditions.
- Search for a range of values and NULL values.
- Search data based on string patterns.



Simple search conditions

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

You'll begin with some simple search conditions.

Overview of search conditions

- Use SELECT statements to retrieve data from database tables.
- Adding a WHERE clause to a SELECT statement limits the data that is retrieved from a table.
- The WHERE clause applies a search condition to each row of the SELECT statement.
- The search condition uses operators to specify the data that the query includes.



The SELECT statement has five main clauses:

- FROM clause (required)
- WHERE clause
- GROUP BY clause
- HAVING clause
- ORDER BY clause

The WHERE clause is used to filter the data that the query returns.

The WHERE clause sets up a search condition that uses a logical test to decide whether a row should be included in a query.

Search conditions compare two values by using an operator to test whether the search condition is met.

If the search condition is met, the query returns the data items from the table.

Search condition in a WHERE clause

Criteria to search for selected values in the table

| country | |
|----------------|-----------|
| Column | Type |
| Code | Character |
| Name | Character |
| Continent | Character |
| Region | Character |
| SurfaceArea | Float |
| IndepYear | Integer |
| Population | Integer |
| LifeExpectancy | Float |
| GNP | Float |
| GNPOld | Float |

Query

```
SELECT continent, surfacearea,  
        population, gnp  
FROM country  
WHERE name = 'Ireland';
```



Output

```
continent surfacearea population  gnp  
-----  
Europe      70273      3775100 75921
```

This query introduces the country table, which contains data about countries from around the world.

The example shows a conditional search that uses the WHERE clause with the equal to (=) comparison operator.

Specifically, this query returns the continent, surface area, population, and gross national product (GNP) data for the country named Ireland.



Operators

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Next, look at the operators that you can use with your searches.

Types of operators

Three types of operators are used in search conditions:

- Arithmetic operators perform mathematical operations.
- Comparison operators compare values between data items.
- Logical operators are used to build compound conditions.

These operators can be used in SELECT, INSERT, UPDATE, and DELETE statements.

The following slides contain examples of each of these operator types.

Arithmetic operators and comparison operators

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The following slides contain examples of arithmetic operators and comparison operators.

Arithmetic operators

| SQL operation | SQL operator |
|---------------------|--------------|
| Addition | + |
| Subtraction | - |
| Multiplication | * |
| Division | / |
| Modulus (remainder) | % |

The SQL arithmetic operators are addition, subtraction, multiplication, division, and modulus (remainder after division).

Comparison operators

| SQL operation | Operation | Description |
|---------------|-----------------------|--|
| = | Equals | Compares two data items to see whether they are equal |
| !=, <> | Not equal | Compares two data items to see whether they are not equal |
| < | Less than | Compares two data items to see whether the value of the data item on the left is less than the value on the right |
| <= | Greater than | Compares two data items to see whether the value of the data item on the left is less than or equal to the value on the right |
| > | Greater than | Compares two data items to see whether the value of the data item on the left is greater than the value on the right |
| >= | Greater than or equal | Compares two data items to see whether the value of the data item on the left is greater than or equal to the value on the right |

The not-equal-to operator has two options: `!=` and `<>`. Which operator you use might be your personal preference, or it might be formally defined according to a company's established coding standards and practices.

Addition arithmetic operator

Query

```
SELECT name, lifeexpectancy, lifeexpectancy+5.5
FROM country
WHERE gnp > 1300000;
```

Arithmetic operator

Adds 5.5 years to each
lifeexpectancy value that is
displayed in the output

Comparison operator

Output

| name | lifeexpectancy | lifeexpectancy+5.5 |
|----------------|----------------|--------------------|
| Germany | 77.4 | 82.9 |
| France | 78.8 | 84.3 |
| United Kingdom | 77.7 | 83.2 |
| Japan | 80.7 | 86.2 |
| United States | 77.1 | 82.6 |

Suppose research shows that the average life expectancy is expected to increase by 5.5 years for countries whose GNP is greater than \$1.3 trillion.

This query calculates the new life expectancy for these countries. It adds 5.5 to the current value for life expectancy and then displays that calculated value in the query result.

Subtraction arithmetic operator

Query

```
SELECT name, continent, population, population-350000
FROM country
WHERE name='United States';
```

Comparison operator

The arithmetic operator subtracts 350,000 from the population value and displays the result in the output.

Output

| name | continent | population | population-350000 |
|---------------|---------------|------------|-------------------|
| United States | North America | 278357000 | 278007000 |

Suppose research shows that the United States population is expected to decline by 350,000 people due to lower birth rates and emigration to other countries.

This query calculates that the new population of the United States will be 278,007,000 people following that decline. It displays that calculated value in the query result.

Multiplication arithmetic operator

Query

```
SELECT name, continent, population, population*1.15
FROM country
WHERE name = 'Malta';
```

Comparison operator

The arithmetic operator increases the population by 15 percent and displays the result in the output.

Output

| name | continent | population | population*1.15 |
|-------|-----------|------------|-----------------|
| Malta | Europe | 380200 | 437230 |

Suppose research shows that Malta expects a 15 percent increase in population as people from around the world move there to live following their retirement.

This query calculates the new population for Malta by multiplying the current value for population by 1.15. It then displays that calculated value in the query result.

Division arithmetic operator

The arithmetic operator divides the population by the surfacearea to display the population density in the output.

Query

```
SELECT name, region, population, surfacearea, population/surfacearea
FROM country
WHERE population/surfacearea > 3000;
```

The population density calculation is also used as a search condition.

Output

| name | region | population | surfacearea | population/surfacearea |
|-----------|-----------------|------------|-------------|------------------------|
| ----- | ----- | ----- | ----- | ----- |
| Gibraltar | Southern Europe | 25000 | 6 | 4166.666667 |
| Hong Kong | Eastern Asia | 6782000 | 1075 | 6308.837209 |
| Macao | Eastern Asia | 473000 | 18 | 26277.777778 |

Suppose you want to find the countries whose population density is more than 3,000 people per square kilometer.

This query divides the total population by the country size (surface area in square kilometers) to calculate the population density.

The WHERE clause uses the same calculation to limit the query results to only those countries whose population density is greater than 3,000 people per square kilometer.

Modulus arithmetic operator

The modulus operator calculates the remainder of the population divided by the surfacearea and displays that remainder in the output.

Query



```
SELECT name, population, surfacearea, population/ surfacearea, population%surfacearea
FROM country
WHERE name = 'San Marino';
```

Output

| name | population | surfacearea | population/surfacearea | population%surfacearea |
|------------|------------|-------------|------------------------|------------------------|
| San Marino | 27000 | 61 | 442.6230 | 38.00 |

This query shows an example of using the modulus function. It calculates the remainder of 38 square kilometers after dividing the population of San Marino by its surface area.

Logical operators

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The following slides contain examples of logical operators.

Logical operators explained

Common SQL logical operators

| SQL operator | Description |
|--------------|---|
| AND | Joins two or more conditions in a WHERE clause. All conditions must be true for data items to be affected by the SQL statement. |
| OR | Joins two or more conditions in a WHERE clause. At least one of the conditions must be true for data items to be affected by the SQL statement. |
| IN | Used for matching on multiple data items in a single WHERE clause by using a list of conditional values. |
| LIKE | Used for matching on multiple data items in a single WHERE clause by using partially matching conditional values referred to as wildcards (denoted by <code>_</code> or <code>%</code>). |
| BETWEEN | Used for matching on multiple data items in a single WHERE clause by specifying a range on matching conditional values. |
| NOT | Is used to reverse the effect of IN, LIKE, and BETWEEN operators. |

This table explains some commonly used SQL logical operators.

AND logical operator

| city | |
|-------------|-----------|
| ID | Integer |
| Name | Character |
| CountryCode | Character |
| District | Integer |
| Population | Integer |

Query

Logical operator

```
SELECT name, district, population
FROM city
WHERE countrycode = 'IND'
AND district = 'Delhi';
```

All **WHERE** clause conditions must be true for an **AND** operator to affect a SQL statement.

Output

| name | district | population |
|------------------|----------|------------|
| Delhi | Delhi | 7206704 |
| New Delhi | Delhi | 301297 |
| Delhi Cantonment | Delhi | 94326 |

This query introduces the city table, which contains data about cities from all around the world.

This query uses the AND operator to return data items for cities in India (IND) whose district is called Delhi.

A city must meet both of these conditions to be included in the query result set.

OR logical operator

Query

```
SELECT name, district, population
FROM city
WHERE district = 'Delhi'
OR district = 'Punjab';
```

Logical operator

At least one **WHERE** clause condition must be true for an **OR** operator to affect a SQL statement.

Output

| name | district | population |
|-----------------------|----------|------------|
| Delhi | Delhi | 7206704 |
| Ludhiana | Punjab | 1042740 |
| Amritsar | Punjab | 708835 |
| Jalandhar (Jullundur) | Punjab | 509510 |
| ... | | |

The query uses the OR operator to return data items for cities with districts called Delhi or Punjab.

A city must meet one or the other of these district conditions to be included in the query result set.

IN logical operator

Query

```
SELECT name, district, population
FROM city
WHERE district IN ('Delhi','Punjab','Kerala');
```

The **IN** operator returns the rows that match the listed values.

Output

| name | district | population |
|---------------------------------|----------|------------|
| Delhi | Delhi | 7206704 |
| Ludhiana | Punjab | 1042740 |
| Amritsar | Punjab | 708835 |
| Cochin (Kochi) | Kerala | 564589 |
| Thiruvananthapuram (Trivandrum) | Kerala | 524006 |
| Jalandhar (Jullundur) | Punjab | 509510 |
| Lahore | Punjab | 5063499 |
| ... | | |

The query uses the IN operator to return data items for cities with districts called Delhi, Punjab, and Kerala.

The query result set will include any city with a name that matches one of the district names in this list.

LIKE logical operator using the % wildcard

Query

```
SELECT name, district, population
FROM city
WHERE district LIKE 'west%';
```

The **wildcard** character % is used with the **LIKE** logical operator to substitute for one or more characters in a string.

Output

| name | district | population |
|-----------|------------------|------------|
| Perth | West Australia | 1096829 |
| Brugge | West Flander | 116246 |
| Cape Town | Western Cape | 2352121 |
| Paarl | Western Cape | 105768 |
| George | Western Cape | 93818 |
| Zamboanga | Western Mindanao | 601794 |
| Bacolod | Western Visayas | 429076 |
| ... | | |

The query uses the LIKE operator with a wildcard to return data items for cities whose district names start with the name West.

The rest of the name can consist of any number and type of characters.

LIKE logical operator using the _ wildcard

Query

```
SELECT countrycode, name
FROM city
WHERE countrycode LIKE '_B_';
```

The **wildcard** character **_** is used with the **LIKE** logical operator to substitute for a single character in a string.

Output

| countrycode | name |
|--------------------------|------------|
| ABW | Oranjestad |
| GBR | London |
| GBR | Birmingham |
| ...(more GBR entries)... | |
| LBN | Beirut |
| LBY | Tripoli |
| LBR | Monrovia |
| ...(more LBY entries)... | |

The query uses the LIKE operator with a wildcard. It returns data items for cities whose country code has the letter B as second character of the three-character code.

The first and third characters can be any value.

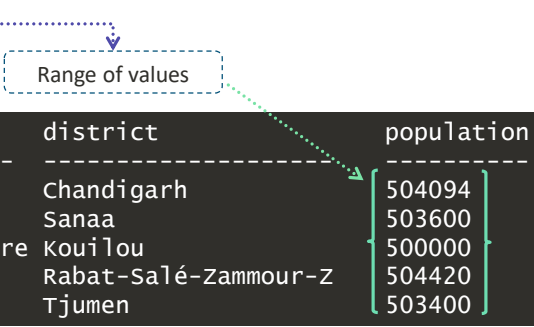
BETWEEN logical operator

Query

```
SELECT countrycode, name, district,  
       population  
FROM city  
WHERE population BETWEEN 500000 AND 505000;
```

The range for a **BETWEEN** operator also includes both specified values.

Output



A dashed blue box labeled "Range of values" has a blue arrow pointing to the "BETWEEN 500000 AND 505000" part of the query. A green dashed arrow points from this box to a green bracket in the output table that encompasses the population values 504094, 503600, 500000, 504420, and 503400.

| countrycode | name | district | population |
|-------------|--------------|----------------------|------------|
| IND | Chandigarh | Chandigarh | 504094 |
| YEM | Sanaa | Sanaa | 503600 |
| COG | Pointe-Noire | Kouilou | 500000 |
| MAR | Salé | Rabat-Salé-Zammour-Z | 504420 |
| RUS | Tjumen | Tjumen | 503400 |

The query uses the BETWEEN operator to return data items for cities whose populations are between 500,000 and 505,000 people.

The first and last value for a BETWEEN clause are included in the range.

Therefore, the query result set will also include any city whose population is exactly 500,000 or 505,000.

NOT logical operator

Query

```
SELECT name, district, population
FROM city
WHERE countrycode = 'CAN'
AND district NOT IN ('Ontario','Alberta')
```

The **NOT** keyword can be used with **IN**, **LIKE**, and **BETWEEN** operators.

Output

| name | district | population |
|-----------|------------------|------------|
| Montréal | Québec | 1016376 |
| Winnipeg | Manitoba | 618477 |
| Vancouver | British Columbia | 514008 |
| ... | | |

The query uses the NOT operator to return data items for cities in Canada, except for cities in Ontario and Alberta.

The query result set will omit any city with a name that matches one of the names in this list.

The NOT operator can be used to modify other SQL operators. For example, you can use the following:

- NOT IN
- NOT BETWEEN
- NOT LIKE

Operators can be used in flexible ways

Query 1

```
SELECT name, district, population
FROM city
WHERE countrycode = 'CAN'
AND district NOT IN ('Ontario','Alberta')
```

Query 2

```
SELECT name, district, population
FROM city
WHERE countrycode = 'CAN'
AND district <> 'Ontario'
AND district <> 'Alberta';
```

Output

These SQL statements use different operators but return the same data items.

| name | district | population |
|-----------|------------------|------------|
| Montréal | Québec | 1016376 |
| Winnipeg | Manitoba | 618477 |
| Vancouver | British Columbia | 514008 |
| ... | | |

Depending on which SQL operators you use, it is possible to achieve the same results by using different SQL statements.

Neither option is correct or incorrect. Which operators you use can be driven by your personal preferences or might be formally defined according to a company's established coding standards and practices.

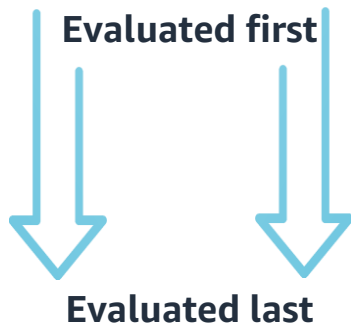
Operator precedence

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

The next slides discuss operator precedence.

Operator precedence explained

SQL operators are evaluated in a defined order in a SQL statement.



Operators enclosed in parentheses

Operators for multiplication or division

Operators for addition or subtraction

Comparison operators

NOT operators used with IN, BETWEEN, and LIKE

AND operators

OR, BETWEEN, IN, and LIKE operators

By default, SQL operators are evaluated in a defined order in a SQL statement.

You might need to have your search criteria evaluated in a different order. You can use parentheses in the WHERE clause to force your criteria to be evaluated in a different order.

Using parentheses to enforce operator precedence: Part 1

```
SELECT name, district, population
FROM city
WHERE countrycode = 'PAK'
AND district = 'Punjab'
OR district = 'Sindh';
```

The query returns data items with countrycode PAK and district of Punjab or district of Sindh.

| name | district | population |
|------------|----------|------------|
| Karachi | Sindh | 9269265 |
| Lahore | Punjab | 5063499 |
| Faisalabad | Punjab | 1977246 |
| Rawalpindi | Punjab | 1406214 |
| Multan | Punjab | 1182441 |
| Hyderabad | Sindh | 1151274 |
| ... | | |

```
SELECT name, district, population
FROM city
WHERE countrycode = 'PAK'
AND district = 'Punjab'
OR district = 'Sindh'
AND population > 1500000
```

An attempt is made to revise the query to exclude cities with populations less than 1.5 million people.

| name | district | population |
|------------|----------|------------|
| Karachi | Sindh | 9269265 |
| Lahore | Punjab | 5063499 |
| Faisalabad | Punjab | 1977246 |
| Rawalpindi | Punjab | 1406214 |
| Multan | Punjab | 1182441 |
| ... | | |

Why does the query result still include cities of less than 1.5 million people?



The first query lists all the data items with countrycode PAK and a district of either Punjab or Sindh.

The second query adds an additional search condition. The second search condition was intended to limit the result set to only those cities with populations of more than 1.5 million people.

However, adding this additional condition to the WHERE clause did not produce the desired result.

This unexpected result happens because, by default, AND conditions are applied to the query before OR conditions.

Therefore, the second query still returns rows for Rawalpindi and Multan, both of which have populations smaller than 1.5 million people.

The next slide shows how to fix this issue by using parentheses.

Using parentheses to enforce operator precedence: Part 2

```
SELECT name, district, population
FROM city
WHERE countrycode = 'PAK'
AND district = 'Punjab'
OR district = 'Sindh'
AND population > 1500000;
```

Why does the query result still include cities of less than 1.5 million people?

| name | district | population |
|------------|----------|------------|
| Karachi | Sindh | 9269265 |
| Lahore | Punjab | 5063499 |
| Faisalabad | Punjab | 1977246 |
| Rawalpindi | Punjab | 1406214 |
| Multan | Punjab | 1182441 |
| ... | | |

```
SELECT name, district, population
FROM city
WHERE countrycode = 'PAK'
AND (district = 'Punjab'
OR district = 'Sindh')
AND population > 1500000;
```

Adding parentheses provides the desired query results.

| name | district | population |
|------------|----------|------------|
| Karachi | Sindh | 9269265 |
| Lahore | Punjab | 5063499 |
| Faisalabad | Punjab | 1977246 |

Note that the query on the left is repeated output from the previous slide for reference.

The goal of the query on the left is to limit the result set to cities with populations of more than 1.5 million people.

However, adding this to the WHERE clause as an additional search condition did not produce the desired result. The query still returns rows for Rawalpindi and Multan, both of which have populations smaller than 1.5 million. This result occurs because SQL processes AND operators before OR operators.

Therefore, the following is the search condition in this query:

- List data items for cities with a district of Punjab. The population size condition is never applied to cities with a district of Punjab. Therefore, you see cities with populations less than 1.5 million in the query result set.
- List data items for cities with a district of Sindh that are also larger than 1.5 million.

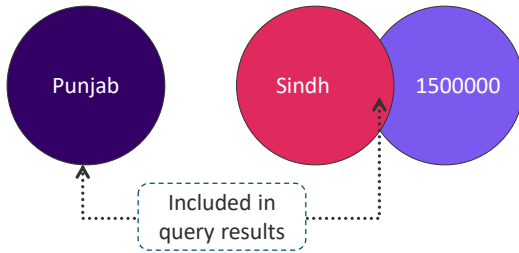
The query on the right shows that adding parentheses to the query can cause the population size condition to be applied after the district name condition.

Enclosing that portion of the query in parentheses causes the OR condition to be evaluated first, followed by the population condition. This adjustment provides the

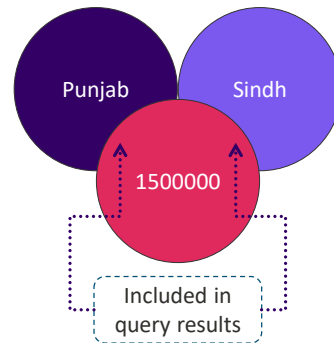
desired query results.

Using parentheses to enforce operator precedence: Part 3

```
SELECT name, district, population
FROM city
WHERE countrycode = 'PAK'
AND district = 'Punjab'
OR district = 'Sindh'
AND population > 1500000;
```



```
SELECT name, district, population
FROM city
WHERE countrycode = 'PAK'
AND (district = 'Punjab'
OR district = 'Sindh')
AND population > 1500000;
```



This visual presentation shows how the use of parentheses around the OR operator in the previous slide helped to achieve the desired query results.

In the first query, all of the data items for Punjab are included in the query results. This result occurs because the population size condition is never applied to cities with a district of Punjab.

In the second query, the query results include all of the data items for Punjab that have more than 1.5 million people. The results also include the data items for Sindh that have more than 1.5 million people.

Activity: Using Operators (1 of 2)



In this activity:

- Consider the following table structure that is used to store data about customer purchases for an online retailer.

| Name | Type |
|---------------|-----------|
| CustomerID | Integer |
| FirstName | Character |
| LastName | Character |
| StreetAddress | Character |
| City | Character |
| StateProvince | Character |
| PostalCode | Integer |
| CountryCode | Character |

Consider the following table structure that is used to store data about customer purchases for an online retailer.

Which SQL statements for the queries can you use?

Activity: Using Operators (2 of 2)



- Using this table, formulate SQL statements for the queries that the instructor describes.

| Name | Type |
|----------------------|-----------|
| CustomerID | Integer |
| FirstName | Character |
| LastName | Character |
| StreetAddress | Character |
| City | Character |
| StateProvince | Character |
| PostalCode | Integer |
| CountryCode | Character |
| EmailAddress | Character |
| LastPurchaseYear | Integer |
| TotalNumberPurchases | Integer |
| TotalCostOfPurchases | Integer |

Develop a query that would return the first name, last name, and email address of customers who answer the following questions:

- Which customers live in a country designated by the country code GDR?
WHERE CountryCode='GDR';
- Which customers live in the city of Chicago in the state of Illinois?
WHERE city = 'Chicago'
AND state = 'Illinois'
- Which customers use a Yahoo email address?
WHERE EmailAddress like '%@yahoo.com'
- Which customers do not use a Yahoo email address?
WHERE EmailAddress not like '%@yahoo.com'
AND EmailAddress is not null
- What are the medium-sized customers, defined as those whose total purchases are more than \$5,000 but less than \$10,000?
WHERE TotalCostOfPurchases between 5000 and 10000

alternatively:

WHERE TotalCostOfPurchases > 4999
AND TotalCostOfPurchases < 10001

6. Which customers have a country code of BRA, made more than 100 purchases last year or spent more than \$20,000 last year, and have a last name of Rodriguez?
WHERE CountryCode='BRA'
AND (TotalNumberPurchases > 100 OR TotalCostOfPurchases > 20000)
AND LastName = 'Rodriguez'

Aliases

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

In previous examples, the returned results sets used column names that matched the column name or field name in the database. In this section, you use aliases to return query results.

Overview of an alias

An alias is used to assign a temporary name to a table or column within a SQL query.

- The alias exists only while the SQL statement is running.
- Aliases are useful for assigning names to obscurely named tables and columns.
- Aliases are also used to assign meaningful names to query output that uses arithmetic SQL operators.
- Aliases can be specified in a SQL statement by using the optional AS keyword.
- If spaces are desired in an alias name, the alias should be defined in quotation marks.

You can use aliases to include a column header of your choosing in a query result set.

In some situations, aliases can make your SQL statements simpler to write and easier to read.

You can use the AS keyword after the column name to create a new column name.

An alias applied to arithmetic operator

```
SELECT name, lifeexpectancy, lifeexpectancy+5.5 AS newlifeexpectancy
FROM country
WHERE gnp > 1300000;
```

Alias is applied here.

| name | lifeexpectancy | newlifeexpectancy |
|----------------|----------------|-------------------|
| Germany | 77.4 | 82.9 |
| France | 78.8 | 84.3 |
| United Kingdom | 77.7 | 83.2 |
| Japan | 80.7 | 86.2 |
| United States | 77.1 | 82.6 |

Suppose research shows that average life expectancy is expected to increase by 5.5 years for countries whose GNP is greater than \$1.3 trillion.

The query lists the new life expectancy for these countries, with the alias **newlifeexpectancy** used for the column name in the query output.

Without this alias, the column in the query output would use the default column name of `lifeexpectancy+5.5`.

Including spaces in an alias

Query

```
SELECT name, lifeexpectancy, lifeexpectancy+5.5 AS 'new life expectancy'  
FROM country  
WHERE gnp > 1300000;
```

Alias

Output

| name | lifeexpectancy | new life expectancy |
|----------------|----------------|---------------------|
| Germany | 77.4 | 82.9 |
| France | 78.8 | 84.3 |
| United Kingdom | 77.7 | 83.2 |
| Japan | 80.7 | 86.2 |
| United States | 77.1 | 82.6 |

Suppose research shows that average life expectancy is expected to increase by 5.5 years for countries whose GNP is greater than \$1.3 trillion.

The query lists the new life expectancy for these countries, with the resulting alias **new life expectancy** (with spaces) used for the column name in the query output.

Without this alias, the column in the query output would use the default column name of `lifeexpectancy+5.5`.

NULL values

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

You'll now look at NULL values.

Overview of NULL values

Databases use **NULL** to represent the absence of value for a data item.

- Because they have no value, **NULL** values cannot be compared to each other by using typical comparison operators.
- Because they have no value, **NULL** values are not equal to one another.
- Use **IS NULL** and **IS NOT NULL** when working with **NULL** values in a **WHERE** clause.
- Tables can be designed so that **NULL** values are not allowed.

A **NULL** is the absence of any value. A zero is not a **NULL** value. A blank space created by pressing the keyboard space bar is not a **NULL** value.

Primary key columns in tables cannot contain **NULL** values.

Non-primary key columns in tables can also be defined as **NOT NULL** if needed.

Example of querying for NULL values

Query

```
SELECT name, lifeexpectancy
FROM country
WHERE lifeexpectancy IS NULL
AND name NOT LIKE '%Island%';
```

Output

| name | lifeexpectancy |
|--------------------------------|----------------|
| Antarctica | NULL |
| French Southern territories | NULL |
| British Indian Ocean Territory | NULL |
| Niue | NULL |
| Pitcairn | NULL |
| Svalbard and Jan Mayen | NULL |
| Tokelau | NULL |
| Holy See (Vatican City State) | NULL |
| Wallis and Futuna | NULL |

IS NULL and IS NOT NULL keywords are used when working with NULL values.

Suppose you want to create some reports related to life expectancy for different countries. You might want to begin by making certain that you have life expectancy data for all the countries in the database.

Countries with no data for life expectancy will have NULL values for this column in the table.

By using the IS NULL operator as a search condition, this query returns data items for countries that do not have values for life expectancy.

NULL values can affect query results

```
SELECT name, lifeexpectancy, lifeexpectancy+1
FROM country
WHERE lifeexpectancy IS NULL
AND name NOT LIKE '%Island%';
```

Adding + 1 to the lifeexpectancy column still returns a **NULL**.

| name | lifeexpectancy | lifeexpectancy+1 |
|--------------------------------|----------------|------------------|
| Antarctica | NULL | NULL |
| French Southern territories | NULL | NULL |
| British Indian Ocean Territory | NULL | NULL |
| Niue | NULL | NULL |
| Pitcairn | NULL | NULL |
| Svalbard and Jan Mayen | NULL | NULL |
| Tokelau | NULL | NULL |
| Holy See (Vatican City State) | NULL | NULL |
| Wallis and Futuna | NULL | NULL |

Because NULL is the absence of a value, any mathematical or comparison operation on NULL values will always return a NULL value.

This property can lead to unexpected query results.

In this query, the + 1 is being added to data items that have a NULL value for lifeexpectancy.

The result of this calculation returns a NULL value for these items.

Discussion: NULL Values



1. What is the purpose of a **NULL** value?
2. Why is it important to search for a **NULL** value in a table?
3. How would you search a table for results that do not contain **NULL**?

1. What is the purpose of a NULL value?

NULL values indicate that no value has been provided for that data item.

2. Why is it important to search for a NULL value in a table?

It is important to search for NULL values in a table. NULL data items can be easily left out of query results when queries on the columns containing NULL values are part of the WHERE clause. This result can cause erroneous results, particularly when arithmetic or comparison operators are applied.

3. How would you search a table for results that do not contain NULL?

To search a table for results that do not contain NULL, include an IS NOT NULL operator in the WHERE clause of the query.

Checkpoint questions



What is the purpose of a condition in a query?



Why should you use aliases in SQL?

1. What is the purpose of a condition in a query?

You can use conditions in a query to filter your search results.

2. Why should you use aliases in SQL?

Aliases are often used to make column names more understandable.

Key takeaways



- A search condition is a logical test that can be applied to a row.
 - The WHERE clause is used to define the search condition.
 - Arithmetic, comparison, and logical operators can be used.
- Naming aliases provides the result set with a column header of your choice.
 - In some situations, the aliases make your SQL statements simpler to write and easier to read.
- NULL values require special handling in SQL statements.
 - NULL is the absence of value.
 - Use IS NULL and IS NOT NULL in queries.

This module includes the following key takeaways:

- A search condition is a logical test that can be applied to a row.
 - The WHERE clause is used to define the search condition.
 - Arithmetic, comparison, and logical operators can be used.
- Naming aliases provides the result set with a column header of your choice.
 - In some situations, the aliases make your SQL statements simpler to write and easier to read.
- NULL values require special handling in SQL statements.
 - NULL is the absence of value.
 - Use IS NULL and IS NOT NULL in queries.



Thank you



© 2022 Amazon Web Services, Inc. or its affiliates. All rights reserved. This work may not be reproduced or redistributed, in whole or in part, without prior written permission from Amazon Web Services, Inc. Commercial copying, lending, or selling is prohibited. Corrections, feedback, or other questions? Contact us at <https://support.aws.amazon.com/#/contacts/aws-training>. All trademarks are the property of their owners.

Thank you for completing this module.