

Data Analytics Training – SQL

Advanced Data Filtering

Advanced Data Filtering

- Using the **AND** Operator - used in a WHERE clause to specify that only rows matching all the specified conditions should be retrieved.

```
SELECT prod_id, prod_price, prod_name  
FROM Products  
WHERE vend_id = 'DLL01' AND prod_price <= 4;
```

Advanced Data Filtering

- Using the **OR** Operator - The OR operator instructs the database management system software to retrieve rows that match either condition.

```
SELECT prod_name, prod_price  
FROM Products  
WHERE vend_id = 'DLL01' OR vend_id = 'BRS01';
```

Advanced Data Filtering

- Combining AND and OR operators – understand order of evaluation
- Use table product to do the practice:
 - I need a list of all products costing \$10 or more made by vendors DLL01 and BRS01
- SQL (like most languages) processes AND operators before OR operators.

```
SELECT prod_name, prod_price
FROM Products
WHERE vend_id = 'DLL01' OR vend_id = 'BRS01'
AND prod_price >= 10;
```

Advanced Data Filtering

- Combining AND and OR operators – Using Parentheses in WHERE Clauses
SELECT prod_name, prod_price
FROM Products
WHERE (vend_id = 'DLL01' OR vend_id = 'BRS01')
AND prod_price >= 10;

Advanced Data Filtering

- Using the IN Operator - used to specify a range of conditions, any of which can be matched
- IN takes a comma-delimited list of valid values, all enclosed within parentheses

```
SELECT prod_name, prod_price  
FROM Products  
WHERE vend_id IN ('DLL01','BRS01');
```

Use Wildcard Filtering

- Wildcard - Special characters used to match parts of a value
- Using the LIKE Operator - To use wildcards in search clauses, the LIKE operator must be used
- Wildcard searching can **only be used with text fields (strings)**, you can't use wildcards to search fields of non-text datatypes

1. The Percent Sign (%) Wildcard – most frequently used wildcard - % means, *match any number of occurrences of any character*

```
SELECT prod_id, prod_name  
FROM Products  
WHERE prod_name LIKE 'Fish%';
```

```
SELECT prod_id, prod_name  
FROM Products  
WHERE prod_name LIKE '%bean bag%';
```


Use Wildcard Filtering

2. The Underscore (_) Wildcard –*match a single character*

```
SELECT prod_id, prod_name  
FROM Products  
WHERE prod_name LIKE '__ inch teddy bear';
```

```
SELECT prod_id, prod_name  
FROM Products  
WHERE prod_name LIKE '% inch teddy bear';
```

Practice 2.1:

Use SCHEMAS(Database) **world** and table **country** to practice:

- List countries in region Eastern Africa or North America or Middle East order by region
- For all countries in region Eastern Asia , select the countries with population > 7000000 or lifeexpectancy > 75
- Identify countries with name beginning with 'A' and ending with 'a'

Create Calculated Fields

Where calculated fields come in?

- Sometimes, the data stored in the table is not exactly what your application needs
- Rather than retrieve the data as it is, what you really want is to retrieve **converted, calculated, or reformatted data** directly from the database
- Calculated fields don't actually exist in database tables. Rather, a calculated field is created on-the-fly **within a SQL SELECT statement**

Concatenating Fields

- Concatenating Fields - Joining values together (by appending them to each other) to form a single long value.
 - *SQL Server support + for concatenation:*
SELECT vend_name + '(' + vend_country + ')'
FROM Vendors
ORDER BY vend_name;
 - MySQL uses CONCAT() to concatenate strings:
*SELECT **concat**(vend_name, ' (' , vend_country, ')')*
FROM Vendors
ORDER BY vend_name;

Concatenating Fields

- Use **Aliases** to name the new calculated column

```
SELECT Concat(vend_name, '(', vend_country, ')')  
AS vend_title  
FROM Vendors  
ORDER BY vend_name;
```

Performing Mathematical Calculations

- Another frequent use for calculated fields is performing mathematical calculations on retrieved data.

```
SELECT prod_id, quantity, item_price, quantity*item_price AS total_sales  
FROM OrderItems  
WHERE order_num = 20008;
```

- SQL Mathematical Operators:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division

Use Data Manipulation Functions

Understand Functions

- Functions perform calculations **on columns!**
- Unlike SQL statements, which for the most part are supported by all DBMSs equally, functions tend to be very DBMS(database management system) specific:
 - DBMS Function Differences:

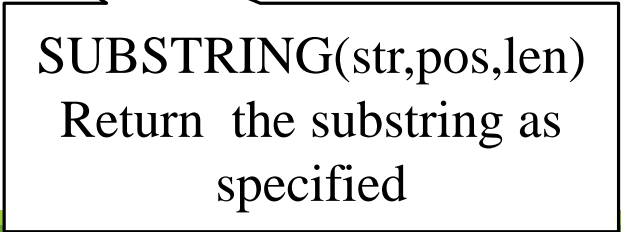
Function	Syntax
Extract part of a string	Access uses MID (). DB2, Oracle, PostgreSQL, and SQLite use SUBSTR (). MariaDB, MySQL and SQL Server use SUBSTRING ().
Datatype conversion	Access and Oracle use multiple functions, one for each conversion type. DB2 and PostgreSQL uses CAST (). MariaDB, MySQL, and SQL Server use CONVERT ().
Get current date	Access uses NOW (). DB2 and PostgreSQL use CURRENT_DATE. MariaDB and MySQL use CURDATE (). Oracle uses SYSDATE. SQL Server use GETDATE (). SQLite uses DATE ().

Different Types of Functions

- **String functions** are used to manipulate strings of text; for example, trimming or padding values and **converting values to upper and lowercase**

```
SELECT vend_name, UPPER(vend_name) AS vend_name_upcase  
FROM Vendors  
ORDER BY vend_name;
```

```
SELECT vend_name, SUBSTRING(vend_name,1,4) AS  
first_4_letters_of_vend_name FROM Vendors  
ORDER BY vend_name;
```



SUBSTRING(str,pos,len)
Return the substring as
specified

Different Types of Functions

- **Date and time functions** are used to manipulate date and time values and to extract specific components from these values; for example, extracting date part from a date value, and checking date validity:

```
SELECT order_num  
FROM Orders  
WHERE YEAR(order_date) = 2012;
```

YEAR() Return
the year

```
SELECT order_num, order_date,  
NOW() as currentdateandtime  
FROM Orders;
```

NOW() Return the current
date and time

```
SELECT order_num, order_date, NOW()  
as currentdateandtime,  
DATEDIFF(curdate(),  
order_date) as dategap  
FROM Orders;
```

DATEDIFF()
Subtract two dates

Different Types of Functions

- **CASE** expression is a very commonly used **control flow function**
- CASE expression can be used to conditionally enter into some logic based on the status of a condition being satisfied
- It is usually used to create a new column
- It is better to make sure each condition is mutually exclusive

```
CASE WHEN [value=compare_value] THEN result  
      [WHEN [value=compare_value] THEN result ...]  
      [ELSE result] END (AS new_column)
```

```
SELECT prod_price,  
       case when prod_price < 6 then 'low price'    else 'high price' end  
from products;
```

Different Types of Functions

- **Numeric functions** are used to perform mathematical operations on numeric data
 - for example, returning absolute numbers and **performing algebraic calculations**
 - Commonly used Numeric Manipulation Functions:

Function	Description
ABS ()	Returns a number's absolute value
COS ()	Returns the trigonometric cosine of a specified angle
EXP ()	Returns the exponential value of a specific number
PI ()	Returns the value of PI
SIN ()	Returns the trigonometric sine of a specified angle
SQRT ()	Returns the square root of a specified number
TAN ()	Returns the trigonometric tangent of a specified angle

Practice 2.2:

Use SCHEMAS(Database) **world** and table **country** to practice:

- Use population/surfacearea to get pop_density and rank pop_density by descending order
- *Bonus: how can I get countries with pop_density > 1000*
- *Create a column called Population_size to segment the country by population size:*
 - *If population < 1 million, then 'small'; if 1 million <= population < 10 million, then 'medium',
if 10 million <= population < 100 million, then 'large; if population >= 100 million, then 'Extra large';*

Then show country name and population_size

- In the table, we found a column called as Code which should be country code, and another column called as Code2. I want to know whether Code2 is just the first 2 letters of Code. Please write query to confirm this. *Hint: use substring to get the first 2 letters of Code, and compare with Code2, if they match with each other, then we can confirm*