Chapter 4: Filtering, Join, and Subquery

Analytics Tensor
Mahesh KC
mahesh.kc@analyticstensor.com
https://analyticstensor.com

Introduction

- Filtering
- Evaluate Condition
- Types of Condition
  - Equality condition
  - Inequality condition
  - Range condition
  - Membership condition
  - Matching condition
- Join
- Subquery
- Wrap-up

Filtering

Filtering is used to filter dataset based on business logic. They are used for multiple purpose such as:-

Creating subset of data.

Removing unwanted or duplicate records.

Updating/Deleting records based on certain condition.

Filtering always refers to *where clause* in SQL statement. For grouping it refer s to *having clause*.

Evaluate Condition

Condition is composed of one or more *expressions* with multiple *operators*. The expression can be:

- Number, like 1, 100
- Column in a table or view
- String literal, like ‘Virginia’, ‘Engineer’
- Built-in function, upper(‘Java’)
- Subquery
- List of expression, like (‘C#’, ‘PHP’, ‘Javascript’)

The operators are:
Comparison Operators, like =, !, <, >, !=, LIKE, IN, BETWEEN
Arithmetic Operators, like +, - , *, /
Boolean Operators, like AND, OR, NOT

Evaluate Condition (cont..)

Table 1: **OR** operator with two conditions

Condition	Result
WHERE true OR true	True
WHERE true OR false	True
WHERE false OR true	True
WHERE false OR false	False

For example:
WHERE state = 'VA' AND purchase_date > '2019-07-12' -- both condition must be true.
WHERE state = 'VA' OR purchase_date > '2019-07-12' -- either one condition must be true.

Table 2: **AND**, **OR** operator with three conditions

Condition	Result
WHERE true AND (true OR true)	True
WHERE true AND (true OR false)	True
WHERE true AND (false OR true)	True
WHERE true AND (false OR false)	False
WHERE false AND (true OR true)	False
WHERE false AND (true OR false)	False
WHERE false AND (false OR true)	False
WHERE false AND (false OR false)	False

Evaluate Condition (cont..)

Table 3: **AND**, **OR** and **NOT** operator with three conditions

Condition	**Result**
WHERE true AND NOT (true OR true)	False
WHERE true AND NOT (true OR false)	False
WHERE true AND NOT (false OR true)	False
WHERE true AND NOT (false OR false)	True
WHERE false AND NOT (true OR true)	False
WHERE false AND NOT (true OR false)	False
WHERE false AND NOT (false OR true)	False
WHERE false AND NOT (false OR false)	False

NOT is the negation operator. It can also be written as ! operator. For example:

WHERE state != 'VA'

If there are more than three conditions then you should use parentheses to make sql statement clear which also helps to make your code easier to read.
For example:
WHERE total_quantity is not NULL AND (state = 'VA' AND purchase_date > '2019-07-12') -- all condition must be true.

Types of Condition

There are various type of conditions to filter data. Such as:

- Equality condition
- Inequality condition
- Range condition
- Membership condition
- Matching condition

All the condition above have their own usages.

Equality Condition

Equality condition equate one expression to another. It has a form of *column_name = expression*

Both value must match to satisfy the equality condition.

For example:

position = ‘Developer’

id = 10115

price= 25

Inequality Condition

Inequality condition is the negation of equality condition. It ensure that two expression are not equal. It has a form of *column_name != expression* or *column_name <> expression*

The value must not match with each other.

For example:

city != ‘Richmond’

dept_no <> 10

amount != 100

Range Condition

Range condition is used to evaluate an expression which satisfies the data within a certain range. It is basically used with numeric and temporal data types.

BETWEEN operator is used to filter data between two range. It requires both lower limit and upper limit of range. For example:

select student_id, fname, lname from student where start_date between ‘2015-01-01’ AND ‘2018-01-01’ ;

Select * from employees where dept_no between 10 and 30;

Membership Condition

Member condition is used to choose a set of finites values rather than just a single values or range of values.

IN operators is used to specify the set of values. This operators is the combination of multiple OR operators.

NOT IN is the reverse of IN operator.

For example:
select * from employees where dept_no in (10, 30, 50)
which can also be written as
select * from employees where dept_no = 10 OR dept_no = 30 or dept_no = 50

Matching Condition

Matching condition is used for partial string matching. It uses wildcard character to match string. Wildcards are used to search character or substring in a string at any positing either beginning, ending, or anywhere within the string.

Table 4: Wildcard characters

Wildcard character	Matches
_	Exactly one character
%	Any number of characters (including 0)

Matching Condition (cont..)

Table 5: String Search Expression

Search expression	Interpretation
S%	Strings beginning with S
%a	Strings ending with a
%jan%	Strings containing the substring 'jan'
_ _i_	Four-character strings with a i in the third position
_ _ _-_ _-_ _ _ _	11-character strings with dashes in the fourth and seventh positions

JOIN

Join is used to retrieve data from multiple tables that shares same column(s). Join allows to combine columns from two or more tables where those tables has at least one matching attributes.

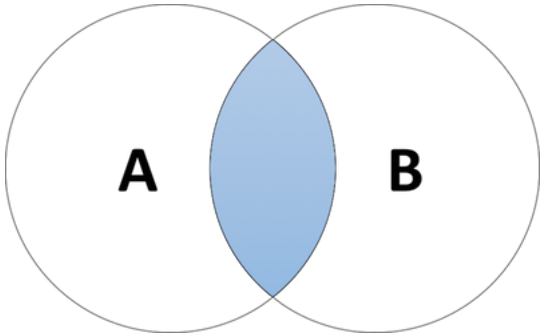
Types of Join

- Inner Join
- Left outer Join
- Right outer Join
- Full outer Join
- Self Join
- Cross Join

```
SELECT A.columns, B.columns
FROM A JOIN B ON A.column_name = B.column_name
```

Inner Join

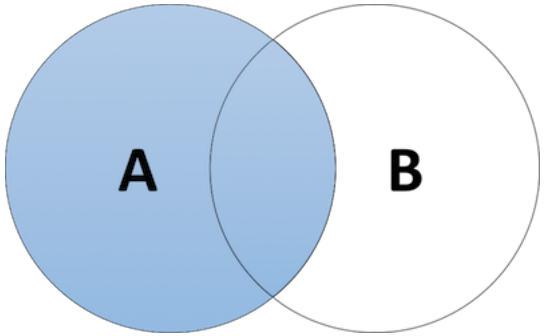
Inner Join returns all the rows from two or more tables that matches the column values in the ON condition.




```
SELECT A.column, B.column
FROM A INNER JOIN B ON A.column_name = B.column_name
```

Left Join

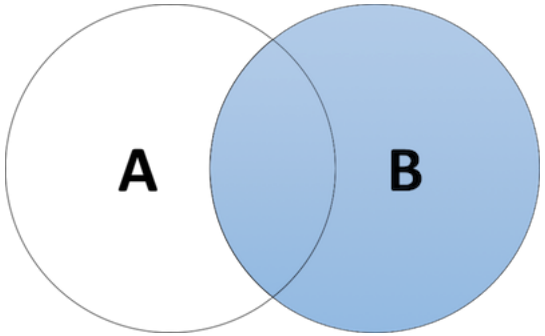
Left Join returns all the rows from the left table specified at ON condition **and only** the rows from right table that matches the join condition.




```
SELECT A.column, B.column
FROM A LEFT JOIN B ON A.column_name = B.column_name
```

Right Join

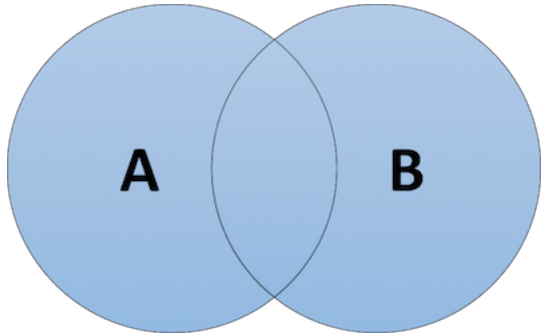
Right Join returns all the rows from the right table specified at ON condition **and only** the rows from left table that matches the join condition.




```
SELECT A.column, B.column
FROM A RIGHT JOIN B ON A.column_name = B.column_name
```

Full Join

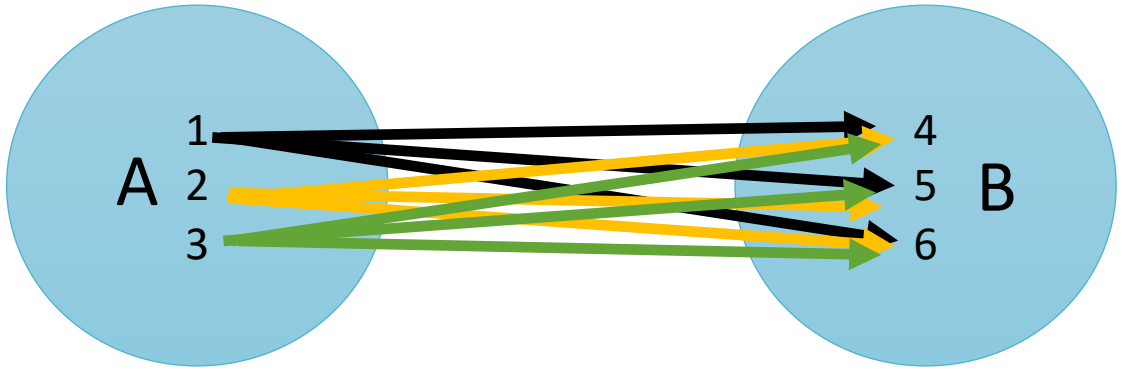
Full Join returns all the rows from both tables even the condition doesn't matches in the ON condition. It is the combination of left and right join. It returns nulls values when the condition doesn't matches and returns values when the condition matches.




```
SELECT A.column, B.column
FROM A FULL JOIN B ON A.column_name = B.column_name
```

Cross Join

Cross Join the cartesian product of two tables. Cross Join returns all the rows from table A multiplied by total number of rows in table B.




```
SELECT A.column, B.column
FROM A CROSS JOIN B
```

Subquery

Subquery is a query contained within a SQL statement. It is a query within a query. Subquery is always enclosed within parentheses. Subquery return dataset that contains:

- Single row with a single column
- Multiple rows with a single column
- Multiple rows and columns

Types of Subquery

- Noncorrelated subquery
- Correlated subquery

Noncorrelated Subquery

Noncorrelated Subquery: The query that does not reference anything from the containing statement is known as noncorrelated subquery. These queries are independent of their containing statements.

Single row and column subquery: Subquery returning single row and column. These type of subquery is also known as scalar subquery. This query appear on either side of condition and uses equality and comparison (=, !>, <, <=, >, >=) operators. The inner query must always return only one record, if it returns more than one record then the error message be prompted with "*Subquery returned more than 1 value. This is not permitted when the subquery follows =, !=, <, <=, >, >= or when the subquery is used as an expression*" OR "*Subquery returns more than 1 row*" based on database vendors.

```
SELECT A.column_1, A.column_2 FROM A WHERE column_1 = (SELECT B.column_name from B where column_name = 'A')
```

Noncorrelated Subquery

Multiple-row, single-column subquery: Subquery returning more than one rows. It does not uses equality or comparison operators instead uses (IN, NOT IN, ALL, ANY) operators. It is used to check if single value exists in a list of values.

IN operator: This operator is used to check if the value defined in the left side exists within the set of values on right side of IN operator. IN operator can also be performed by using multiple comparison or equality operator. The converse of IN is **NOT IN** operator.

```
SELECT A.column_1, A.column_2 FROM A WHERE column_1 != ALL (SELECT B.column_name from B where column_name like 'Sr%')
```

ALL operator: ALL operator is used to compare value between a single values and every values in a set. It need to be used with comparison operators (=, !=, <, >) in conjunction with the ALL operator. It returns TRUE if and only if all of the subqueries meet the condition. Make sure to check null values on set.

ANY Operator: ANY operator is used to compare values between set of values like ALL operator but returns true if the inner query contains at least one matching row unlike all the rows in ALL operator.

```
SELECT A.column_1, A.column_2 FROM A WHERE column_1 > ANY (SELECT B.column_name from B where column_name like 'Sr%')
```

Noncorrelated Subquery (cont..)

Multicolumn subquery: The query defined earlier only returns a single column and one or more rows. Multicolumn subquery returns more than one column. It is used to compare the result with two or more columns. Rows returned from the subquery are evaluated in the main query in pair-wise or nonpairwise comparison.

```
SELECT A.column_1, A.column_2 FROM A WHERE (column_1, column_2 IN (SELECT B.column_name,
B.column_name from B where column_name is NOT NULL)
```

Correlated Subquery

Correlated Subquery: Correlated subquery is dependent on its containing statement from which it references one or more columns. A correlated subquery is executed once for each candidate row. It uses equality condition to reference the inner query. Range condition can also be used to filter the rows. For each rows in outer query, all the rows are evaluated once in the inner query.

Exists operator: Exists operator is used to identify if a relationship exists or not. The result of Exists operator is boolean value (True/False). It can be used in SELECT, INSERT, UPDATE or DELETE statement. The converse of EXISTS is NOT EXISTS.

- If the subquery returns at least one record then the exists clause will be evaluated true and the condition will satisfy.
- If the subquery does not return any records then the exists clause will be evaluated false and the condition will not satisfy.

```
SELECT A.column_1, A.column_2 FROM A WHERE EXISTS (SELECT 1 from B where B.column_name = A.column_1)
```

Wrap-up

- Important points.
- Q & A.

Assignment-5

Describe the types of subquery with an example. Use at least two tables to illustrate the SQL query.

Use the references below:

https://www.ibm.com/support/knowledgecenter/en/ssw_ibm_i_74/sqlp/rbafysubquery.htm
https://docs.microsoft.com/en-us/sql/relational-databases/performance/subqueries?view=sql-server-2017#fundamentals
https://www.akadia.com/services/sqlsrv_subqueries.html