SELECT - aggregate

Quick Reference

Syntax

```
... { AVG( [DISTINCT] col )
    | MAX( [DISTINCT] col|sql_exp )
    | MIN( [DISTINCT] col|sql_exp )
    | SUM( [DISTINCT] col|sql_exp )
    | COUNT( DISTINCT col|sql_exp )
    | COUNT( * )
    | COUNT(*) } ...
```

Effect

Aggregate expression for

- Specifies `col_spec` for a column of the results set in the `SELECT` list in the `SELECT` statement,
- Specifies an operand in the `HAVING` clause.

In an aggregate expression, a column `col` of a data source or an SQL expression `sql_exp` is specified as an argument of one of the aggregate functions shown here. Each aggregate function calculates a single value from the values of the column or from the results of the SQL expression and from multiple rows as follows. Here, the addition `DISTINCT` excludes duplicate values from the calculation:

- **`AVG( [DISTINCT] col )`**

  Determines the average value of the contents of column `col` in the results set or in the current group. The data type of the column must be numeric. The data types INT8, DF16_RAW, and DF34_RAW plus the obsolete types DF16_SCL and DF34_SCL are not allowed. The data type of the result for decimal floating point numbers is the corresponding data type (DF16_DEC or DF34_DEC) and otherwise FLTP. SQL expressions cannot be specified as arguments.

- **`MAX( [DISTINCT] col|sql_exp )`**
  `MIN( [DISTINCT] col|sql_exp )`

  Determine the maximum value or minimum value of the values of the column `col` or of the results of the SQL expression `sql_exp` in the results set or the current group. The data type of the result is the external data type of the corresponding column or of the result of the SQL expression. A column with any data type can be specified for `col`. The result of an SQL expression `sql_exp` must be a numeric type (except for the types for decimal floating point numbers).

- **`SUM( [DISTINCT] col|sql_exp )`**

  Determines the total of the content of the column `col` or of the results of the SQL expression `sql_exp` in the results set or the current group. The data type of the result is the external data type of the corresponding column or of the result of the SQL expression. The data type of the column `col` must be a numeric type. The data types DF16_RAW and DF34_RAW and the obsolete types DF16_SCL and DF34_SCL are not allowed. The result of an SQL expression `sql_exp` must be a numeric type (except for the types for decimal floating point numbers).

- **`COUNT( DISTINCT col|sql_exp )`**

  Determines the number of distinct values in the column `col` or of the results of the SQL expression `sql_exp` in the results set or the current group. The data type of the result is INT4. A column with any data type can be specified for `col`. An SQL expression `sql_exp` can be a result with any type (except for the types for decimal floating point numbers).

- **`COUNT( * )`**
  `COUNT( * )`

  Determines the number of rows in the results set or in the current group. No column identifier is specified in this case.

  - If `COUNT( * )` or `COUNT(*)` is specified in a `SELECT` list with other columns or together with a `GROUP BY` clause, the data type of the result is INT4 and no numbers greater than 2147483647 can be determined.
  - If `COUNT( * )` or `COUNT(*)` is specified as the only column and the `GROUP BY` clause is not specified, the internal data type of the result is INT8 and numbers up to +9223372036854775807 can be determined. The system field `sy-dbcnt` is set to the value -1 in results outside of the value range of the type `i`.

If the value of an aggregate expression is too large for the target area, an exception is raised. More specifically, a target object of the type `p` or `decfloat34` must be specified if a standalone function `COUNT( * )` or `COUNT(*)` expects a value greater than the value range of INT4.

If the argument of an aggregate function has the null value, it is ignored when the function is evaluated. The result is a null value only if all the rows in the column in question contain a null value.

Notes

- The database platform determines whether an overflow occurs if the result of an aggregate function exceeds its value range. On some database platforms, intermediate results outside the value range are allowed. The overflow behavior of SQL expressions, on the other hand, is platform-independent. An aggregate expression with an SQL expression raises an exception on every platform in the case of an overflow, even if a corresponding result of the aggregate function would not raise an exception on every platform.
- If the aggregate function `SUM` is used for columns of type DF16_DEC, it is best to use a target field of data type `decfloat34` to avoid overflows.
- If required, columns can be specified as `col` using a path expression for associations of a CDS view.
- If SQL expressions are specified as arguments of aggregate expressions, the syntax check is performed in a strict mode, which handles the statement more strictly than the regular syntax check.

Example

Determines the number of airlines flying to New York.

```
DATA count TYPE i.

SELECT COUNT( DISTINCT carrid )
       FROM spfli
       WHERE cityto = 'NEW YORK'
       INTO @count.
```

Example

Returns the flight date, the number of passengers, and the average and maximum luggage weight of all Lufthansa flights with the flight number 0400.

```
TYPES: BEGIN OF wa,
        fldate LIKE sbook-fldate,
        count  TYPE i,
        avg    TYPE p DECIMALS 2,
        max    TYPE p DECIMALS 2,
      END OF wa.
DATA  itab TYPE TABLE OF wa WITH EMPTY KEY.

SELECT fldate, COUNT( * ), AVG( luggweight ), MAX( luggweight )
       FROM sbook
       WHERE carrid = 'LH' AND
             connid = '0400'
       GROUP BY fldate
       INTO TABLE @itab.

cl_demo_output=>display( itab ).
```

Example

See SQL Expressions, Use in Aggregate Expressions .