* Tools for database management that offer built-in functions for performing operations on data within the database itself.
* That when working with large datasets, you may save time by using built-in functions rather than first retrieving the data into your application and then executing functions on the retrieved data.
* You can use sub-queries to form more powerful queries than otherwise.
* You can use a sub-select expression to evaluate some built-in aggregate functions like the average function.
* Derived tables or table expressions are sub-queries where the outer query uses the results of the sub-query as a data source.

|  |  |  |  |
| --- | --- | --- | --- |
| **Command** | **Syntax (MySQL/DB2)** | **Description** | **Example (MySQL/DB2)** |
| COUNT | SELECT COUNT(column\_name) FROM table\_name WHERE condition; | COUNT function returns the number of rows that match a specified criterion. | SELECT COUNT(dep\_id) FROM employees; |
| AVG | SELECT AVG(column\_name) FROM table\_name WHERE condition; | AVG function returns the average value of a numeric column. | SELECT AVG(salary) FROM employees; |
| SUM | SELECT SUM(column\_name) FROM table\_name WHERE condition; | SUM function returns the total sum of a numeric column. | SELECT SUM(salary) FROM employees; |
| MIN | SELECT MIN(column\_name) FROM table\_name WHERE condition; | MIN function returns the smallest value of the SELECTED column. | SELECT MIN(salary) FROM employees; |
| MAX | SELECT MAX(column\_name) FROM table\_name WHERE condition; | MAX function returns the largest value of the SELECTED column. | SELECT MAX(salary) FROM employees; |
| ROUND | SELECT ROUND(2number, decimals, operation) AS RoundValue; | ROUND function rounds a number to a specified number of decimal places. | SELECT ROUND(salary) FROM employees; |
| LENGTH | SELECT LENGTH(column\_name) FROM table; | LENGTH function returns the length of a string (in bytes). | SELECT LENGTH(f\_name) FROM employees; |
| UCASE | SELECT UCASE(column\_name) FROM table; | UCASE function displays the column name in each table in uppercase. | SELECT UCASE(f\_name) FROM employees; |
| LCASE | SELECT LCASE(column\_name) FROM table; | LCASE function displays the column name in each table in lowercase. | SELECT LCASE(f\_name) FROM employees; |
| DISTINCT | SELECT DISTINCT column\_name FROM table; | DISTINCT function is used to display data without duplicates. | SELECT DISTINCT UCASE(f\_name) FROM employees; |
| DAY | SELECT DAY(column\_name) FROM table | DAY function returns the day of the month for a given date. | SELECT DAY(b\_date) FROM employees where emp\_id = 'E1002'; |
| CURRENT\_DATE | SELECT CURRENT\_DATE; | CURRENT\_DATE is used to display the current date. | SELECT CURRENT\_DATE; |
| DATEDIFF() | SELECT DATEDIFF(date1, date2); | DATEDIFF() is used to calculate the difference between two dates or time stamps. The default value generated is the difference in number of days. | SELECT DATEDIFF(CURRENT\_DATE, date\_column) FROM table; |
| FROM\_DAYS() | SELECT FROM\_DAYS(number\_of\_days); | FROM\_DAYS() is used to convert a given number of days to YYYY-MM-DD format. | SELECT FROM\_DAYS(DATEDIFF(CURRENT\_DATE, date\_column)) FROM table; |
| DATE\_ADD() | SELECT DATE\_ADD(date, INTERVAL n type); | DATE\_ADD() is used to calculate the date after lapse of mentioned number of units of date type, i.e. if n=3 and type=DAY, the result is a date 3 days after what is mentioned in date column. The type valiable can also be months or years. | SELECT DATE\_ADD(date, INTERVAL 3 DAY);; |
| DATE\_SUB() | SELECT DATE\_SUB(date, INTERVAL n type); | DATE\_SUB() is used to calculate the date prior to the record date by mentioned number of units of date type, i.e. if n=3 and type=DAY, the result is a date 3 days before what is mentioned in date column. The type valiable can also be months or years. | SELECT DATE\_SUB(date, INTERVAL 3 DAY);; |
| Subquery | SELECT column\_name [, column\_name ] FROM table1 [, table2 ] WHERE column\_name OPERATOR (SELECT column\_name [, column\_name ] FROM table1 [, table2 ] [WHERE]) | Subquery is a query within another SQL query and embedded within the WHERE clause.  A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved. | SELECT emp\_id, f\_name, l\_name, salary FROM employees where salary < (SELECT AVG(salary) FROM employees);   SELECT \* FROM ( SELECT emp\_id, f\_name, l\_name, dep\_id FROM employees) AS emp4all;     SELECT \* FROM employees WHERE job\_id IN (SELECT job\_ident FROM jobs); |
| Implicit Inner Join | SELECT column\_name(s) FROM table1, table2 WHERE table1.column\_name = table2.column\_name; | Implicit Inner Join combines two or more records but displays only matching values in both tables. Inner join applies only the specified columns. | SELECT \* FROM employees, jobs where employees.job\_id = jobs.job\_ident; |
| Implicit Cross Join | SELECT column\_name(s) FROM table1, table2; | Implicit Cross Join is defined as a Cartesian product where the number of rows in the first table is multiplied by the number of rows in the second table. | SELECT \* FROM employees, jobs; |