

Expressions

An expression is a combination of one or more values, operators and SQL functions that evaluate to a value. These SQL EXPRESSIONs are like formulae and they are written in query language. You can also use them to query the database for a specific set of data.

Consider the basic syntax of the SELECT statement as follows —

```
SELECT column1, column2, columnN  
FROM table_name  
WHERE [CONDITION|EXPRESSION];
```

There are different types of SQL expressions, which are mentioned below —

- Boolean
- Numeric
- Date

Let us now discuss each of these in detail.

Boolean Expressions

SQL Boolean Expressions fetch the data based on matching a single value. Following is the syntax —

```
SELECT column1, column2, columnN  
FROM table_name  
WHERE SINGLE VALUE MATCHING EXPRESSION;
```

Consider the CUSTOMERS table having the following records —

SQL> SELECT * FROM CUSTOMERS;					
ID	NAME	AGE	ADDRESS	SALARY	
1	Ramesh	32	Ahmedabad	2000.00	
2	Khilan	25	Delhi	1500.00	
3	kaushik	23	Kota	2000.00	
4	Chaitali	25	Mumbai	6500.00	
5	Hardik	27	Bhopal	8500.00	
6	Komal	22	MP	4500.00	
7	Muffy	24	Indore	10000.00	

```

+-----+
7 rows in set (0.00 sec)

```

The following table is a simple example showing the usage of various SQL Boolean Expressions –

```

SQL> SELECT * FROM CUSTOMERS WHERE SALARY = 10000;
+-----+-----+-----+-----+-----+
| ID | NAME | AGE | ADDRESS | SALARY |
+-----+-----+-----+-----+-----+
| 7 | Muffy | 24 | Indore | 10000.00 |
+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)

```

Numeric Expression

These expressions are used to perform any mathematical operation in any query. Following is the syntax –

```

SELECT numerical_expression as OPERATION_NAME
[FROM table_name
WHERE CONDITION] ;

```

Here, the numerical expression is used for a mathematical expression or any formula. Following is a simple example showing the usage of SQL Numeric Expressions –

```

SQL> SELECT (15 + 6) AS ADDITION
+-----+
| ADDITION |
+-----+
| 21 |
+-----+
1 row in set (0.00 sec)

```

Date Expressions

Date Expressions return current system date and time values –

```

SQL> SELECT CURRENT_TIMESTAMP;
+-----+
| Current_Timestamp |
+-----+
| 2009-11-12 06:40:23 |
+-----+
1 row in set (0.00 sec)

```

Another date expression is as shown below –

```

SQL> SELECT GETDATE();
+-----+

```

```
| GETDATE |
+-----+
| 2009-10-22 12:07:18.140 |
+-----+
1 row in set (0.00 sec)
```

CREATE DATABASE

The SQL **CREATE DATABASE** statement is used to create a new SQL database.

A. Syntax

The basic syntax of this CREATE DATABASE statement is as follows —

```
CREATE DATABASE DatabaseName;
```

Always the database name should be unique within the RDBMS.

Example

If you want to create a new database <testDB>, then the CREATE DATABASE statement would be as shown below —

```
SQL> CREATE DATABASE testDB;
```

Make sure you have the admin privilege before creating any database. Once a database is created, you can check it in the list of databases as follows —

```
SQL> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| AMROOD |
| TUTORIALSPOINT |
| mysql |
| orig |
| test |
| testDB |
+-----+
7 rows in set (0.00 sec)
```

DROP DATABASE

The SQL **DROP DATABASE** statement is used to drop an existing database in SQL schema.

A. Syntax

The basic syntax of DROP DATABASE statement is as follows –

DROP DATABASE DatabaseName;

Always the database name should be unique within the RDBMS.

Example

If you want to delete an existing database <testDB>, then the DROP DATABASE statement would be as shown below –

```
SQL> DROP DATABASE testDB;
```

NOTE – Be careful before using this operation because by deleting an existing database would result in loss of complete information stored in the database.

Make sure you have the admin privilege before dropping any database. Once a database is dropped, you can check it in the list of the databases as shown below –

```
SQL> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| AMROOD          |
| TUTORIALSPPOINT  |
| mysql           |
| orig            |
| test           |
+-----+
6 rows in set (0.00 sec)
```

USE Statement

When you have multiple databases in your SQL Schema, then before starting your operation, you would need to select a database where all the operations would be performed.

The SQL **USE** statement is used to select any existing database in the SQL schema.

A. Syntax

The basic syntax of the USE statement is as shown below —

```
USE DatabaseName;
```

Always the database name should be unique within the RDBMS.

Example

You can check the available databases as shown below —

```
SQL> SHOW DATABASES;
+-----+
| Database |
+-----+
| information_schema |
| AMROOD         |
| TUTORIALSPOINT  |
| mysql          |
| orig           |
| test          |
+-----+
6 rows in set (0.00 sec)
```

CREATE TABLE

Creating a basic table involves naming the table and defining its columns and each column's data type.

The SQL **CREATE TABLE** statement is used to create a new table.

A. Syntax

The basic syntax of the CREATE TABLE statement is as follows —

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
```

```

.....
columnN datatype,
PRIMARY KEY( one or more columns )
);

```

CREATE TABLE is the keyword telling the database system what you want to do. In this case, you want to create a new table. The unique name or identifier for the table follows the CREATE TABLE statement.

Then in brackets comes the list defining each column in the table and what sort of data type it is. The syntax becomes clearer with the following example.

A copy of an existing table can be created using a combination of the CREATE TABLE statement and the SELECT statement. You can check the complete details at [Create Table Using another Table](#).

Example

The following code block is an example, which creates a CUSTOMERS table with an ID as a primary key and NOT NULL are the constraints showing that these fields cannot be NULL while creating records in this table —

```

SQL> CREATE TABLE CUSTOMERS (
    ID      INT                NOT NULL,
    NAME    VARCHAR (20)      NOT NULL,
    AGE     INT                NOT NULL,
    ADDRESS CHAR (25) ,
    SALARY  DECIMAL (18, 2),
    PRIMARY KEY (ID)
);

```

You can verify if your table has been created successfully by looking at the message displayed by the SQL server, otherwise you can use the **DESC** command as follows —

```

SQL> DESC CUSTOMERS;
+-----+-----+-----+-----+-----+-----+
| Field | Type          | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| ID    | int(11)       | NO   | PRI |         |       |
| NAME  | varchar(20)   | NO   |     |         |       |
| AGE   | int(11)       | NO   |     |         |       |
| ADDRESS | char(25)     | YES  |     | NULL    |       |

```

SALARY	decimal(18,2)	YES		NULL	
5 rows in set (0.00 sec)					

DROP TABLE

Now, you have CUSTOMERS table available in your database which you can use to store the required information related to customers.

The SQL **DROP TABLE** statement is used to remove a table definition and all the data, indexes, triggers, constraints and permission specifications for that table.

NOTE — You should be very careful while using this command because once a table is deleted then all the information available in that table will also be lost forever.

A. Syntax

The basic syntax of this DROP TABLE statement is as follows —

DROP TABLE table_name;

Example

Let us first verify the CUSTOMERS table and then we will delete it from the database as shown below —

```
SQL> DESC CUSTOMERS;
```

Field	Type	Null	Key	Default	Extra
ID	int(11)	NO	PRI		
NAME	varchar(20)	NO			
AGE	int(11)	NO			
ADDRESS	char(25)	YES		NULL	
SALARY	decimal(18,2)	YES		NULL	

5 rows in set (0.00 sec)

This means that the CUSTOMERS table is available in the database, so let us now drop it as shown below.

```
SQL> DROP TABLE CUSTOMERS;
Query OK, 0 rows affected (0.01 sec)
```

Now, if you would try the DESC command, then you will get the following

error —

```
SQL> DESC CUSTOMERS;  
ERROR 1146 (42S02): Table 'TEST.CUSTOMERS' doesn't exist
```

INSERT INTO

The SQL **INSERT INTO** Statement is used to add new rows of data to a table in the database.

Syntax

There are two basic syntaxes of the INSERT INTO statement which are shown below.

```
INSERT INTO TABLE_NAME (column1, column2, column3,...columnN)  
VALUES (value1, value2, value3,...valueN);
```

Here, column1, column2, column3,...columnN are the names of the columns in the table into which you want to insert the data.

You may not need to specify the column(s) name in the SQL query if you are adding values for all the columns of the table. But make sure the order of the values is in the same order as the columns in the table.

The **SQL INSERT INTO** syntax will be as follows —

```
INSERT INTO TABLE_NAME VALUES  
(value1,value2,value3,...valueN);
```

You can create a record in the CUSTOMERS table by using the second syntax as shown below.

```
INSERT INTO CUSTOMERS  
VALUES (7, 'Muffy', 24, 'Indore', 10000.00 );
```

All the above statements would produce the following records in the CUSTOMERS table as shown below.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00

4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

SELECT

The SQL **SELECT** statement is used to fetch the data from a database table which returns this data in the form of a result table. These result tables are called result-sets.

A. Syntax

The basic syntax of the SELECT statement is as follows —

SELECT column1, column2, columnN FROM table_name;

Here, column1, column2... are the fields of a table whose values you want to fetch. If you want to fetch all the fields available in the field, then you can use the following syntax.

```
SELECT * FROM table_name;
```

Example

Consider the CUSTOMERS table having the following records —

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

UPDATE

The SQL **UPDATE** Query is used to modify the existing records in a table. You can use the WHERE clause with the UPDATE query to update the selected rows, otherwise all the rows would be affected.

A. Syntax

The basic syntax of the UPDATE query with a WHERE clause is as follows —

UPDATE table_name

SET column1 = value1, column2 = value2..., columnN = valueN

WHERE [condition];

You can combine N number of conditions using the AND or the OR operators.

Example

Consider the CUSTOMERS table having the following records —

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following query will update the ADDRESS for a customer whose ID number is 6 in the table.

```
SQL> UPDATE CUSTOMERS
SET ADDRESS = 'Pune'
WHERE ID = 6;
```

Now, the CUSTOMERS table would have the following records —

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	Pune	4500.00
7	Muffy	24	Indore	10000.00

DELETE

The SQL DELETE Query is used to delete the existing records from a table.

You can use the WHERE clause with a DELETE query to delete the selected rows, otherwise all the records would be deleted.

A. Syntax

The basic syntax of the DELETE query with the WHERE clause is as follows —

```
DELETE FROM table_name  
WHERE [condition];
```

You can combine N number of conditions using AND or OR operators.

Example

Consider the CUSTOMERS table having the following records —

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00
5	Hardik	27	Bhopal	8500.00
6	Komal	22	MP	4500.00
7	Muffy	24	Indore	10000.00

The following code has a query, which will DELETE a customer, whose ID is 6.

```
SQL> DELETE FROM CUSTOMERS  
WHERE ID = 6;
```

Now, the CUSTOMERS table would have the following records.

ID	NAME	AGE	ADDRESS	SALARY
1	Ramesh	32	Ahmedabad	2000.00
2	Khilan	25	Delhi	1500.00
3	kaushik	23	Kota	2000.00
4	Chaitali	25	Mumbai	6500.00

	5		Hardik		27		Bhopal		8500.00	
	7		Muffy		24		Indore		10000.00	
+	-----	+	-----	+	-----	+	-----	+	-----	+