**Module - 3 (RDBMS)**

Introduction To RDBMS(DB2)

What is Db2

IBM Db2 is a family of related data management products, including relational database servers, developed and marketed by IBM.

Since the 1970s, IBM has developed a complete family of database servers, started on mainframe platforms such as Virtual Machine (VM), Virtual Storage Extended (VSE), and Multiple Virtual Storage(MVS).

In 1983, IBM released Db2 for MVS version 1. IBM used Db2 to indicate a shift from a hierarchical database such as Information Management System (IMS) to the new relational database.

Since then IBM continued to develop Db2 on mainframe and distributed platforms. The distributed platforms are referred to as open system platforms such as Linux, UNIX, and Windows.

In 1996, IBM released Db2 Universal Database for distributed platforms, or Db2 UDB version 5. This was the first Db2 version that IBM designed to optimize for the web.

Since the DB2 UDB version 5, IBM continued delivering a number of releases of Db2 over the years with lots of innovations such as pureXML that supports XML documents in 2006. In 2009, IBM introduced the Db2 SQL compatibility feature that makes it easy for customers to migrate from other relational database products to Db2.

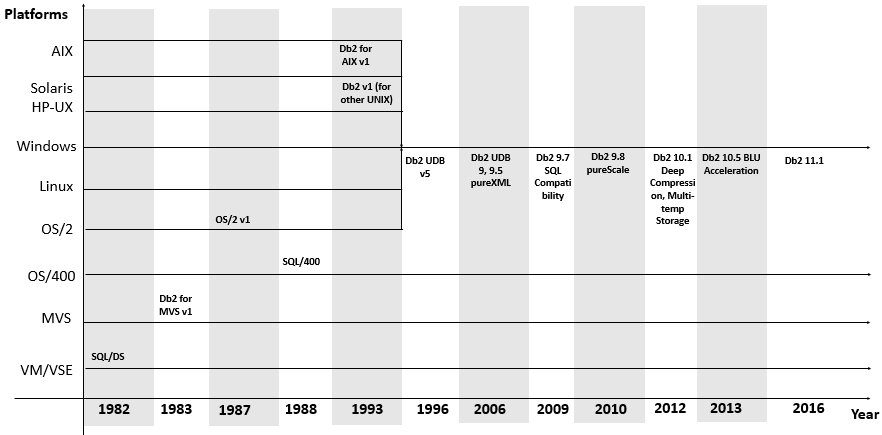
In 2012, IBM released Db2 10.1 with several features that help customers reduce operational costs. A notable feature was multi-temperature storage. For example, you can classify the data as hot and cold. The hot data is the data that you access often, which will be resided in the fastest and more expensive hardware. While the cold data is the data you rarely access, which will be stored in slower and cheaper devices.

In 2013, IBM released Db2 10.4 that introduced BLU acceleration, a memory-based column store which leverages parallel vector processing power, dynamic memory capability, and advanced storage disk technology.

In 2016, IBM released Db2 11.1 that delivered key innovation which reduces the total cost of ownership. Db2 11.1 also enhanced analytics and increased availability, reliability, and security for your business-critical applications.

In mid-2017, IBM re-branded its DB2 and dashDB product offerings to Db2.

The following picture illustrates the history of Db2:



# DB2 Sample Database

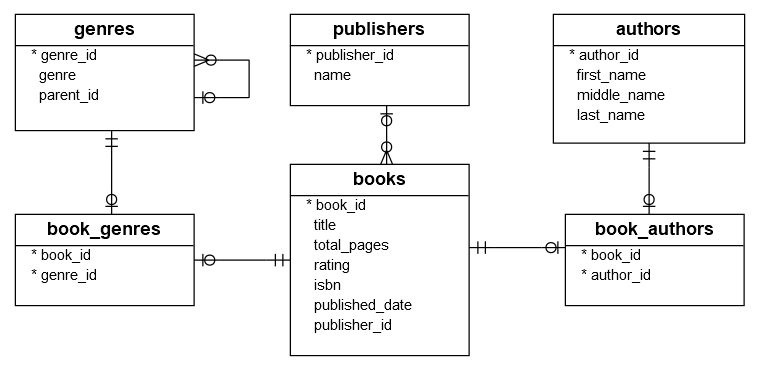
**Summary**: in this tutorial, we will introduce you to the Books DB2 sample database for practicing with DB2 tutorials.

## Introduction to Books DB2 sample database

The Books database is a simple sample database designed for learning and practicing DB2. It consists of six tables:

* books table stores book data including title, total pages, rating, ISBN, and published date.
* publishers table stores publisher names.
* authors table stores books’ authors.
* book\_authors table stores the relationship between books and authors. A book can be written by one or more authors, and one author may have one or many books.
* genres table stores book’s genres. Genres data is hierarchical which is specified by values in the parent\_id column
* book\_genres table stores the relationship between books and genres. A book may belong to one or more genres and a genre may have one or many books.

The following database diagram illustrates the tables and their relationships:



## Database Tables

### Table publishers

The  publishers table has two columns that store publisher identification and name.



|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE publishers(    publisher\_id INT GENERATED BY DEFAULT AS IDENTITY NOT NULL,    name         VARCHAR(255) NOT NULL,    PRIMARY KEY(publisher\_id)  ); |

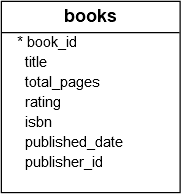


### Table books

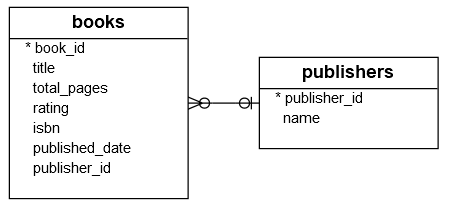
The books table has 7 columns that store book identification, title, total pages, ISBN, published date, and the identification of the publisher. Each book belongs to a publisher and a publisher may have one or many books. If the value in the publisher column is NULL, it means the publisher is unknown at the time of recording the book.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | CREATE TABLE books(    book\_id        INT GENERATED BY DEFAULT AS IDENTITY NOT NULL,    title          VARCHAR(255) NOT NULL,    total\_pages    INT NULL,    rating         DECIMAL(4, 2) NULL,    isbn           VARCHAR(13) NULL,    published\_date DATE NULL,    publisher\_id   INT NULL,    PRIMARY KEY(book\_id),    CONSTRAINT fk\_publisher      FOREIGN KEY(publisher\_id)      REFERENCES publishers(publisher\_id)  ); |



The following picture illustrates the relationship between  books and publishers tables:

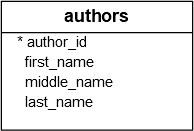


### Table authors

The  authors table has 4 columns that store author identification, first name, middle name, and last name.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE TABLE authors(    author\_id   INT GENERATED BY DEFAULT AS IDENTITY NOT NULL,    first\_name  VARCHAR(100) NOT NULL,    middle\_name VARCHAR(50) NULL,    last\_name   VARCHAR(100) NULL,    PRIMARY KEY(author\_id)  ); |

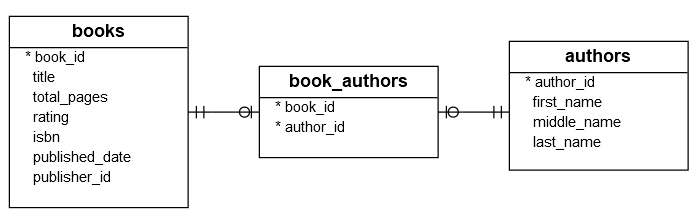


### Table book\_authors



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | CREATE TABLE book\_authors (    book\_id   INT NOT NULL,    author\_id INT NOT NULL,    PRIMARY KEY(book\_id, author\_id),    CONSTRAINT fk\_book      FOREIGN KEY(book\_id)      REFERENCES books(book\_id) ON DELETE CASCADE,    CONSTRAINT fk\_author      FOREIGN KEY(author\_id)      REFERENCES authors(author\_id) ON DELETE CASCADE  ); |

Each author has one or many books while each book is written by one or multiple authors. The relationship between books and authors is many to many as described in the following picture:



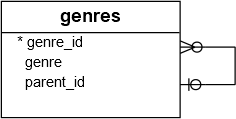
### Table genres

The genres table has three columns that store genre identification, genre, and the relationship between genres.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | CREATE TABLE genres (    genre\_id  INT GENERATED BY DEFAULT AS IDENTITY NOT NULL,    genre     VARCHAR(255) NOT NULL,    parent\_id INT NULL,    PRIMARY KEY(genre\_id),    CONSTRAINT fk\_parent      FOREIGN KEY(parent\_id) REFERENCES genres(genre\_id)  ); |

The following picture shows the genres table:



### Table book\_genres

The book\_genres table stores the relationship between books and genres by using two columns: book\_id and genre\_id.

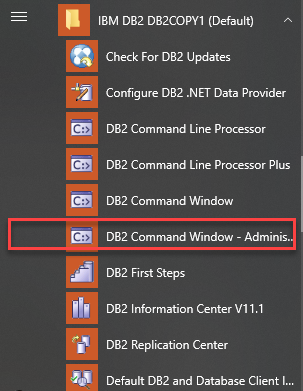


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | CREATE TABLE book\_genres(    book\_id  INT NOT NULL,    genre\_id INT NOT NULL,    PRIMARY KEY(book\_id, genre\_id),    CONSTRAINT fk\_book      FOREIGN KEY(book\_id)      REFERENCES books(book\_id) ON DELETE CASCADE,    CONSTRAINT fk\_genre      FOREIGN KEY(genre\_id)      REFERENCES genres(genre\_id) ON DELETE CASCADE  ); |

# Create Db2 Sample Database

**Summary**: in this tutorial, you will learn how to create a new database in Db2 and load the sample database into the database server.

### Step 1. Launch the Db2 command window



### Step 2. Create the Books database

From the DB2 command window, type db2 command, you will see the following command line:



|  |  |
| --- | --- |
| 1 | db2 => |

Use the CREATE DATABASE statement to create the Books database:



|  |  |
| --- | --- |
| 1 | db2 => CREATE DATABASE books |

It will take a while to create the database. Once, you see the following message:



|  |  |
| --- | --- |
| 1 | DB20000I  The CREATE DATABASE command completed successfully. |

It means you have successfully created a new database.

To list all databases in the current instance, you use the list database directory command:



|  |  |
| --- | --- |
| 1 | db2 => list database directory |

Here is the output:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | System Database Directory    Number of entries in the directory = 2    Database 1 entry:    Database alias                       = BOOKS  Database name                        = BOOKS  Local database directory             = C:  Database release level               = 14.00  Comment                              =  Directory entry type                 = Indirect  Catalog database partition number    = 0  Alternate server hostname            =  Alternate server port number         =    Database 2 entry:    Database alias                       = SAMPLE  Database name                        = SAMPLE  Local database directory             = C:  Database release level               = 14.00  Comment                              =  Directory entry type                 = Indirect  Catalog database partition number    = 0  Alternate server hostname            =  Alternate server port number         = |

There are two databases. The SAMPLE database that we created after [installing DB2](https://www.db2tutorial.com/getting-started/install-db2/) database server and the Books database that we have created.

### Step 3. Download Books sample database script files

Click the following link to download the Books sample database script. You must extract the zip file into a directory e.g., D:\bookdb

[Download the Books Sample Database](https://www.db2tutorial.com/wp-content/uploads/2019/06/books.zip)

There are three files:

1. create.sql for creating tables
2. data.sql for inserting data into the tables
3. drop.sql for deleting all tables

### Step 4. Load Books sample database

First, use quit command to exit the session:



|  |  |
| --- | --- |
| 1 | db2 => quit |

you are now back to the BIN directory.

Next, connect to the books database using the db2admin user:



|  |  |
| --- | --- |
| 1 | > db2 connect to books user db2admin using password |

Note that you must replace the password to your password that you provided during installing the Db2 database server.

Then, use the following command to run the create.sql script to create tables:



|  |  |
| --- | --- |
| 1 | > db2 -stvf d:\bookdb\create.sql |

Verify if all commands completed successfully.

After that, use the following command to execute the data.sql script to insert data into the tables:



|  |  |
| --- | --- |
| 1 | > db2 -stvf d:\bookdb\data.sql |

It will take a while to complete all commands in the file.

Finally, use the following command to double check if data is loaded successfully:



|  |  |
| --- | --- |
| 1 | >db2 select count(\*) author\_count from authors |

If you see the following output, then congratulation, you have successfully created the books database:



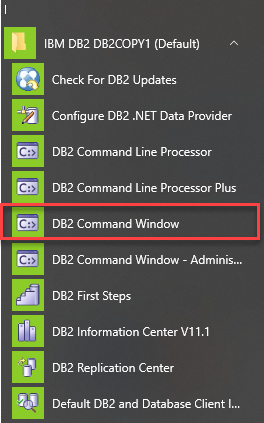
|  |  |
| --- | --- |
| 1  2  3  4  5 | AUTHOR\_COUNT  ------------          1388    1 record(s) selected. |

# Connecting to a Database in Db2 Database Server

**Summary**: in this tutorial, you will learn how to connect to a database in the Db2 database server using various client tools.

## Connecting to a database using the Db2 Command Line tool

First, launch the DB2 command window:



Next, type db2 command:



|  |  |
| --- | --- |
| 1 | C:\Program Files\IBM\SQLLIB\BIN>db2 |

you will see the following command line processor for Db2 client appears:



|  |  |
| --- | --- |
| 1 | db2 => |

Then, use the CONNECT command to connect to a specific database e.g., the Books database:



|  |  |
| --- | --- |
| 1 | db2 => connect to books user db2admin using your\_password |

This command allowed us to connect to the Books database using the db2admin user with the password. You must provide the correct password for the db2admin user.

Here is the output:



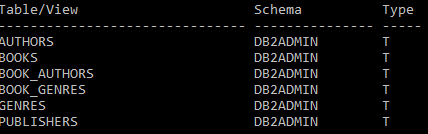
|  |  |
| --- | --- |
| 1  2  3  4  5 | Database Connection Information    Database server        = DB2/NT64 11.1.4.4  SQL authorization ID   = DB2ADMIN  Local database alias   = BOOKS |

After that, you can issue any SQL statement or Db2 command to interact with the Books database. For example, the following command lists all tables in the Books database:



|  |  |
| --- | --- |
| 1 | db2 => list tables |

The following picture shows the output:



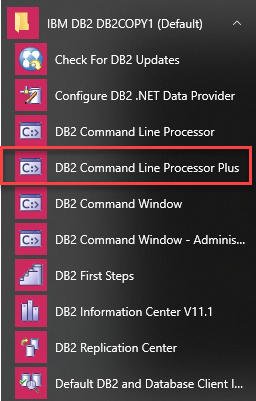
Finally, exit the session by using the quit command:



|  |  |
| --- | --- |
| 1  2 | db2 => quit  DB20000I  The QUIT command completed successfully. |

## Connecting to a database using the Db2 Command Line Plus (CLP) tool

DB2 provides a tool which is similar to SQLPlus in Oracle called Db2 Command Line Plus (or CLP) tool, which can be accessed as shown in the following screenshot:



Once you launch the CLP tool, you will see a command window with the following command:



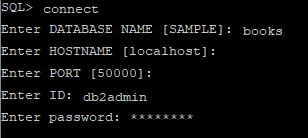
|  |  |
| --- | --- |
| 1 | SQL> |

To connect to a database, you use the connect command as follows:

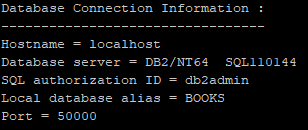


|  |  |
| --- | --- |
| 1 | SQL> connect |

The CLP tool will request for the detailed information including the database, hostname, port, user (ID), and password. Press enter to accept the default value supplied in the () or provide your own values:



If you provided the correct information, you should see the following message:



From now on, you can start issuing the command to interact with the Books database. For example, you can use the following statement to get the number of books:



|  |  |
| --- | --- |
| 1 | SQL> SELECT COUNT(\*) FROM books; |

Here is the output:



|  |  |
| --- | --- |
| 1  2  3 | BOOK\_COUNT  -----------         1226 |

To close the CLP tool, you use the quit command:



|  |  |
| --- | --- |
| 1 | SQL> quit |

# Db2 SELECT

**Summary**: in this tutorial, you will learn how to query data from one or more column of a table by using the Db2 SELECT statement.

## Introduction to Db2 SELECT statement

The SELECT statement queries data from one or more tables in a database. It is one of the most complex SQL statements in Db2. In this tutorial, we will focus on using the SELECT statement to query data from a single table.

Here is the simplest form of the SELECT statement:

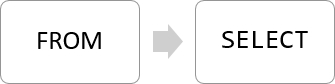


|  |  |
| --- | --- |
| 1  2  3  4 | SELECT      select\_list  FROM      table\_name; |

In this syntax:

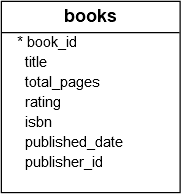
* First, specify a list of comma-separated columns or expressions in the SELECT clause.
* Then, specify the table from which you want to query data in the FROM clause.

When evaluating the SELECT statement, Db2 evaluates the FROM clause first and then the SELECT clause:



## Db2 SELECT statement examples

Let’s take the books table from the [sample database](https://www.db2tutorial.com/getting-started/db2-sample-database/) for the demonstration.



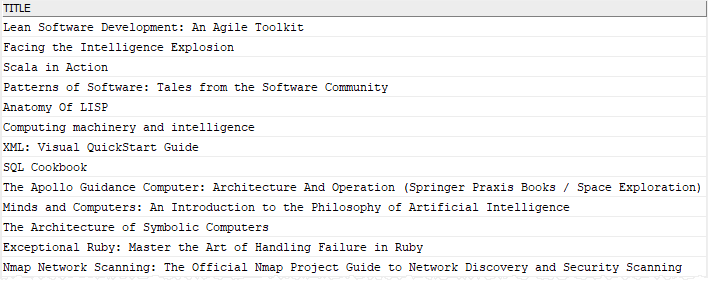
### 1) Using Db2 SELECT statement to query data from one column example

The following statement returns titles of all rows in the books table:



|  |  |
| --- | --- |
| 1  2  3  4 | SELECT      title  FROM      books; |

Here is the partial result set:



In this statement, we specified the books table in the FROM clause and the title column in the SELECT clause.

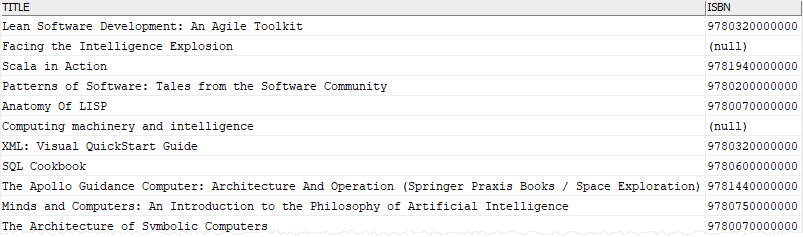
### 2) Using Db2 SELECT statement to query data from multiple columns example

This example returns the title and ISBN of all books from the books table:



|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT      title,      isbn  FROM      books; |

The following picture shows the partial output:



In this example, the select list has two columns: title and ISBN, therefore, we use a comma as a separator between them.

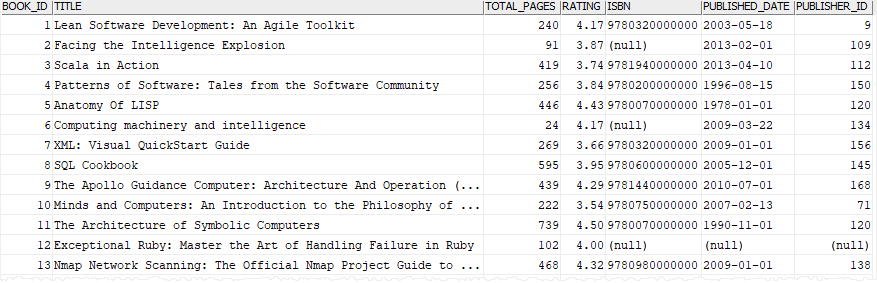
### 3) Using Db2 SELECT statement to query data from all columns of a table example

To query data from all columns of a table, you list all columns in the SELECT clause:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | SELECT      book\_id,      title,      total\_pages,      rating,      isbn,      published\_date,      publisher\_id  FROM      books; |

The output is as follows:



Another way to query data from all columns of a table is to use an asterisk (\*) shorthand as shown in the following query:



|  |  |
| --- | --- |
| 1  2  3  4 | SELECT    \*  FROM      books; |

The (\*) is a shorthand for all columns.

### 4) Using SELECT statement without referencing a table

If you want to call a function or evaluate an expression using the SELECT statement, you can use the sysibm.sysdummy1 table:



|  |  |
| --- | --- |
| 1  2  3  4 | SELECT     expression  FROM     sysibm.sysdummy1; |

The sysibm.sysdummy1 is a special in-memory table which you can use to evaluate expressions or discover Db2 registers.

For example, this statement returns the current date:



|  |  |
| --- | --- |
| 1  2  3  4 | SELECT     CURRENT\_DATE  FROM     sysibm.sysdummy1; |

# Db2 ORDER BY

**Summary**: in this tutorial, you will learn how to use the Db2 ORDER BY clause to sort the result of a query in ascending or descending order.

## Introduction to Db2 ORDER BY clause

When you use the [SELECT](https://www.db2tutorial.com/db2-basics/db2-select/) statement to query data from a table, the order of rows in the result set is unspecified.

To sort the result set by values in one or more columns, you use the ORDER BY clause.

The ORDER BY clause is an optional clause of the SELECT statement. It always appears at the end of the SELECT statement as follows:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      select\_list  FROM      table\_name  ORDER BY      expression1 [ASC | DESC],      expression2 [ASC | DESC],      ...; |

In this syntax:

* First, specify expression1, expression2, etc., that can be columns or expressions by which you want to sort the result set.
* Second, use ASC to sort the result set in ascending order (from low to high) and DESC to sort the result set in descending order (from high to low).

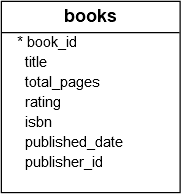
The ASC or DESC is optional. If you skip it, Db2 will use ASC by default.

When evaluating the SELECT statement with an ORDER BY clause, Db2 evaluates the clauses in the following order: FROM, SELECT, and ORDER BY. In other words, Db2 always evaluates the ORDER BY clause at last.



## Db2 ORDER BY clause examples

We will use the books table from the [sample database](https://www.db2tutorial.com/getting-started/db2-sample-database/) to demonstrate the ORDER BY clause.



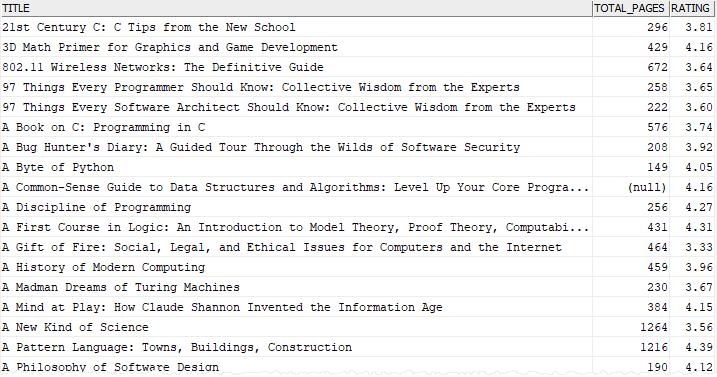
### 1) Using Db2 ORDER BY clause to sort a result set by values in one column example

This example returns titles, total pages, and ratings of all books sorted by titles in ascending order:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      title,      total\_pages,      rating  FROM      books  ORDER BY      title; |

Here is the partial output:



In this example, we did not specify ASC or DESC after the title column in the ORDER BY clause, therefore, Db2 sorted books by titles alphabetically.

### 2) Using Db2 ORDER BY clause to sort a result set by values in two columns example

The following example sorts books by the number of pages in descending order and titles in ascending order:

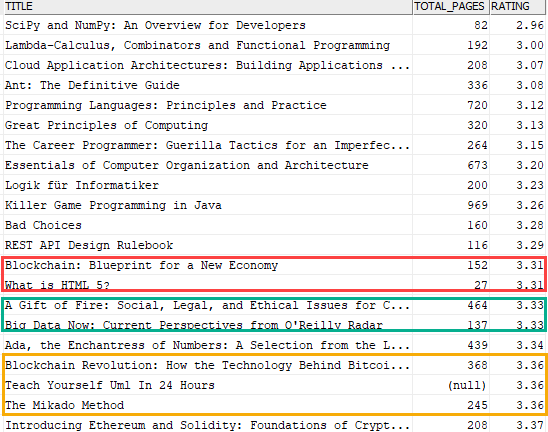


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      title,      total\_pages,      rating  FROM      books  ORDER BY      rating DESC,      title; |

In this example:

* First, sort all books by ratings from high to low to make an initial result set sorted by ratings.
* Second, sort the sorted result set by titles. It means that if two or more books that have the same ratings, the ORDER BY clause will sort those books by titles in ascending order.

The following picture shows the partial output:



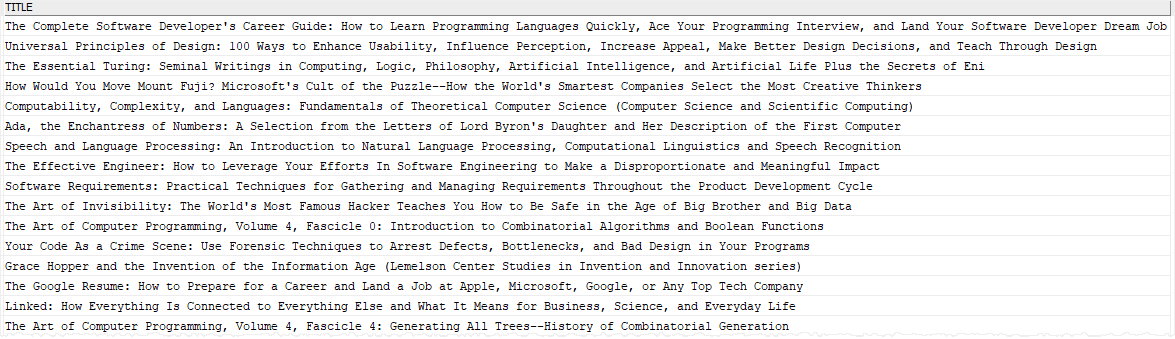
### 3) Using Db2 ORDER BY clause to sort a result by results of an expression

The LENGTH() function returns the length of a string. This example uses the ORDER BY clause to sort the books by the length of the titles from long to short:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT      title  FROM      books  ORDER BY      LEN(title) DESC; |

Here is the partial result set:



### 4) Using Db2 ORDER BY clause to sort a result set by ordinal positions of columns

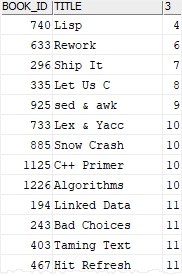
It is possible to use the ordinal positions of columns in the select list in the ORDER BY clause for sorting the result set.

The following example sorts the books by the lengths of their titles. However, instead of using the expression LENGTH(title) explicitly in the ORDER BY clause, it uses the ordinal positions of the expression:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      book\_id,      title,      LENGTH(title)  FROM      books  ORDER BY      3; |

Here is the partial output:



It is a good practice to avoid the ordinal positions of columns in the ORDER BY clause. Because using the ordinal positions of columns in the ORDER BY clause makes the query difficult to maintain and may cause bugs if you forget to change the ordinal positions in the ORDER BY clause when you modify the select list.

## Db2 ORDER BY clause with NULL values

The NULL values are special. They are the markers indicating missing values. When you sort a list of values that consists of NULL values, you can specify whether to treat NULL values as the lowest or highest values by using the NULLS FIRST or NULLS LAST option:



|  |  |
| --- | --- |
| 1 | ORDER BY expression [NULLS FIRST | NULLS LAST] |

The following query returns the title and ISBN of books. It treats NULL values as the lowest values:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT     title,     isbn  FROM     books  ORDER BY     isbn NULLS FIRST; |

The following query treats NULL values as the highest values by using the NULLS LAST option in the ORDER BY clause:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT     title,     isbn  FROM     books  ORDER BY     isbn NULLS LAST; |

# Db2 WHERE

**Summary**: in this tutorial, you will learn how to use the Db2 WHERE clause to specify the search condition for rows returned by a query.

## Introduction to Db2 WHERE clause

The WHERE clause is an optional clause of the [SELECT](https://www.db2tutorial.com/db2-basics/db2-select/) statement. The WHERE clause specifies the search condition for the rows returned by the SELECT statement.

The following shows the syntax of the WHERE clause in the SELECT statement:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      select\_list  FROM      table\_name  WHERE      search\_condition  ORDER BY      sort\_expression; |

In this syntax, the search\_condition like a filter that defines a condition for the returned rows. The rows that cause the search\_condition evaluate to true will be included in the result set.

The search\_condition may consist of one or many logical expressions that evaluate to true, false, or unknown. The logical expressions in the WHERE clause are also known as predicates.

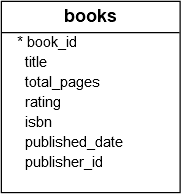
The WHERE clause appears after the FROM clause and before the [ORDER BY](https://www.db2tutorial.com/db2-basics/db2-order-by/) clause. When evaluating the SELECT statement, Db2 evaluates the clauses in the following sequence: FROM, WHERE, SELECT, and ORDER BY.



Besides the SELECT statement, the WHERE clause are used in the UPDATE or DELETE statement to specify rows to be updated or deleted.

## Db2 WHERE clause examples

We’re going to use the books table from the [sample database](https://www.db2tutorial.com/getting-started/db2-sample-database/) to demonstrate the WHERE clause:



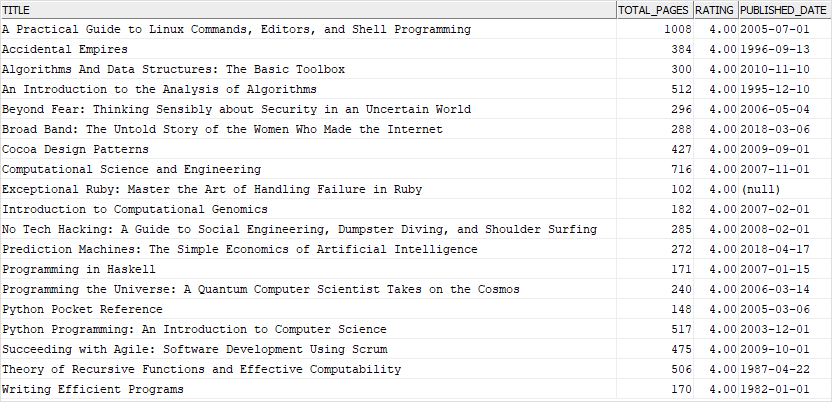
### 1) Using simple equality operator in Db2 WHERE clause example

This query uses the WHERE clause to find books whose ratings are 4:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      title,      total\_pages,      rating,      published\_date  FROM      books  WHERE      rating = 4  ORDER BY      title; |

Here are the output:



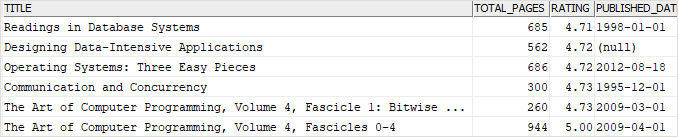
### 2) Using comparison operators in the Db2 WHERE clause example

The following query uses the WHERE clause to return books whose ratings are greater than 4.7 and less than or equal to 5:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | SELECT      title,      total\_pages,      rating,      published\_date  FROM      books  WHERE rating > 4.7          AND rating <= 5  ORDER BY      rating,      title; |

The following picture shows the result set:

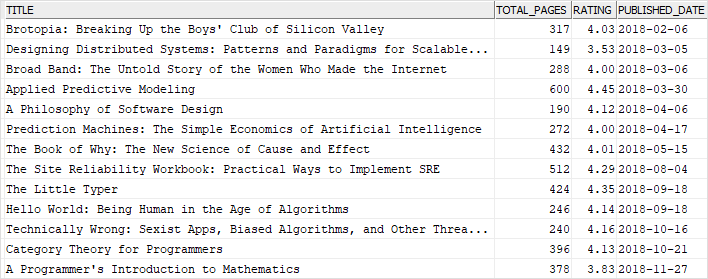


### 3) Using Db2 WHERE clause to find rows with the value between two values

The following statement uses the [BETWEEN](https://www.db2tutorial.com/db2-basics/db2-between/) operator in the WHERE clause to find books whose published dates are between Jan 01, 2018 and December 31, 2018:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      title,      total\_pages,      rating,      published\_date  FROM      books  WHERE      published\_date BETWEEN '2018-01-01' AND '2018-12-31'  ORDER BY      published\_date; |

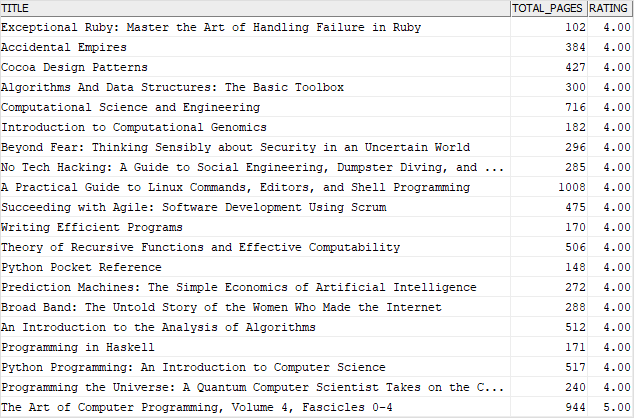


### 4) Using Db2 WHERE clause to find rows that have a value in a list of values

The following example uses the [IN](https://www.db2tutorial.com/db2-basics/db2-in/) operator in the WHERE clause to find books whose rating is 4 or 5



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      title,      total\_pages,      rating  FROM      books  WHERE rating IN (4, 5)  ORDER BY      rating; |



### 5) Using Db2 WHERE clause to find rows whose values contain a string

This example uses the [LIKE](https://www.db2tutorial.com/db2-basics/db2-like/) operator in the WHERE clause to find books whose title contains the string 'SQL':



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | SELECT      title,      total\_pages,      rating,      published\_date  FROM      books  WHERE title LIKE '%SQL%'  ORDER BY      title; |

# Db2 SELECT DISTINCT

**Summary**: in this tutorial, you will learn how to use the Db2 SELECT DISTINCT to prevent duplicate rows returned by a query.

## Introduction to Db2 SELECT DISTINCT

Sometimes, you want to select distinct values from one or more columns of a table. To do this, you use the DISTINCT keyword in the [SELECT](https://www.db2tutorial.com/db2-basics/db2-select/) clause as follows:



|  |  |
| --- | --- |
| 1  2  3  4 | SELECT      DISTINCT column\_name  FROM      table\_name; |

The DISTINCT keyword appears after the SELECT keyword but before any column or expression in the select list. The query above returns distinct values in the column\_name from the table\_name.

If you have multiple column names listed after the DISTINCT keyword like the following query:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT      DISTINCT          column\_name1,          column\_name2, ...  FROM      table\_name; |

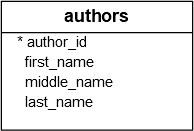
The DISTINCT keyword is applied to all columns. It means that the query will use the combination of values in all columns to evaluate the distinction.

If you want to select distinct values of some columns in the select list, you should use the [GROUP BY](https://www.db2tutorial.com/db2-basics/db2-group-by/) clause.

In case a column contains multiple NULL values, DISTINCT will keep only one NULL in the result set.

## Db2 SELECT DISTINCT examples

We will use the authors table from the [sample database](https://www.db2tutorial.com/getting-started/db2-sample-database/) for the demonstration.



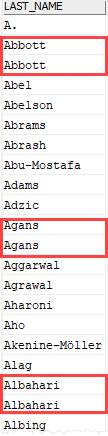
### 1) Using Db2 SELECT DISTINCT with one column

The following query returns all last names of authors from the authors table:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT      last\_name  FROM      authors  ORDER BY      last\_name; |

Here is the result set:



As clearly shown in the output, we had many authors with the same last name e.g., Abbott, Agans, and Albahari.

To get unique author’s last names, you add the DISTINCT keyword as shown in the following query:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT DISTINCT      last\_name  FROM      authors  ORDER BY      last\_name; |

Here is the output:



As you can see clearly from the output, the DISTINCT operator keeps one value for each group of duplicates.

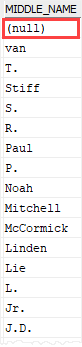
### 2) Using Db2 SELECT DISTINCT with NULL values

The middle\_name column of the authors table contains many rows with NULL values. When we apply the DISTINCT to the middle\_name column, only one instance of NULL is included in the result set as shown in the result set of the following query;



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT DISTINCT      middle\_name  FROM      authors  ORDER BY      middle\_name DESC; |

Here is the output:



### 3) Using Db2 SELECT DISTINCT with multiple columns

Let’s set up a new table for the demonstration.

First, [create a new table](https://www.db2tutorial.com/db2-basics/db2-create-table/) named book\_inventories:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE TABLE book\_inventories  (      book\_id  INT NOT NULL,      store\_id INT NOT NULL,      quantity INT NOT NULL,      PRIMARY KEY(book\_id, store\_id)  ); |

Second, [insert](https://www.db2tutorial.com/db2-basics/db2-insert/) some rows into the table:

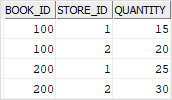


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | INSERT INTO book\_inventories(book\_id, store\_id, quantity)  VALUES(100, 1, 15);  INSERT INTO book\_inventories(book\_id, store\_id, quantity)  VALUES(100, 2, 20);  INSERT INTO book\_inventories(book\_id, store\_id, quantity)  VALUES(200, 1, 25);  INSERT INTO book\_inventories(book\_id, store\_id, quantity)  VALUES(200, 2, 30); |

Third, [query](https://www.db2tutorial.com/db2-basics/db2-select/) data from the book\_inventories table:



|  |  |
| --- | --- |
| 1  2 | SELECT \*  FROM book\_inventories; |



If you use the DISTINCT keyword on the book\_id, you will get two values 100 and 200 as shown in the output of the following query:



|  |  |
| --- | --- |
| 1  2  3  4 | SELECT DISTINCT      book\_id  FROM      book\_inventories; |



|  |  |
| --- | --- |
| 1  2  3  4 | BOOK\_ID  -----------  100  200 |

However, when you add the store\_id column to the query as follows:



|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT DISTINCT      book\_id,      store\_id  FROM      book\_inventories; |



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | BOOK\_ID     STORE\_ID  ----------- -----------  100         1  100         2  200         1  200         2 |

It returns the distinct values of both book\_id and store\_id.

# Db2 AND

**Summary**: in this tutorial, you will learn how to use the Db2 AND operator to combine multiple Boolean expressions.

## Introduction Db2 AND operator

The AND operator is a logical operator that combines two Boolean expressions or predicates. You use the AND operator to specify that a search must satisfy both conditions.

The following illustrates the AND operator syntax:



|  |  |
| --- | --- |
| 1 | boolean\_expression1 AND boolean\_expression2 |

In this syntax, the boolean\_expression1 and boolean\_expression2 evaluate to true, false, and unknown.

This table displays the result when combining true, false, and unknown values using the AND operator:

|  | **TRUE** | **FALSE** | **UNKNOWN** |
| --- | --- | --- | --- |
| **TRUE** | TRUE | FALSE | UNKNOWN |
| **FALSE** | FALSE | FALSE | FALSE |
| **UNKNOWN** | UNKNOWN | FALSE | UNKNOWN |

If you use both AND and OR operators in an expression, Db2 evaluates the AND operator first. However, you can change the order of evaluation using parentheses.

To negate the AND operator, you use the NOT operator as follows:

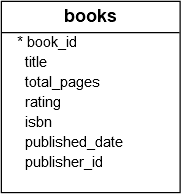


|  |  |
| --- | --- |
| 1 | NOT (boolean\_expression1 AND boolean\_expression2) |

The AND operator is often used in the [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause to form the search condition for the [SELECT](https://www.db2tutorial.com/db2-basics/db2-select/), [UPDATE](https://www.db2tutorial.com/db2-basics/db2-update/), and [DELETE](https://www.db2tutorial.com/db2-basics/db2-delete/) statement.

## Db2 AND operator examples

Let’s use the books table from the sample database for the demo.



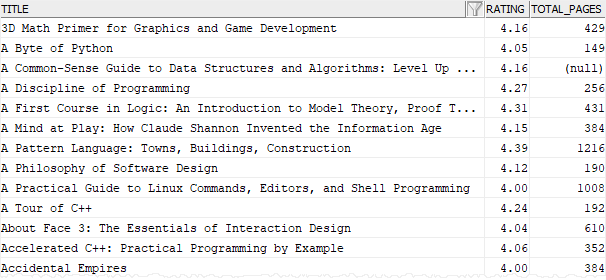
### 1) Db2 AND operator example

This example uses the AND operator to find the books whose ratings are between 4 and 5:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | SELECT      title,      rating,      total\_pages  FROM      books  WHERE rating >= 4          AND rating <= 5  ORDER BY      title; |

Here is the result set:



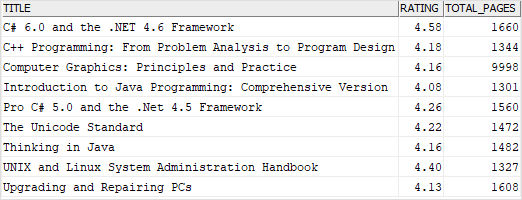
### 2) Using multiple Db2 AND operators example

This example uses two AND operators to find books whose ratings are between 4 and 5 and the number of pages is greater than 1300:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | SELECT      title,      rating,      total\_pages  FROM      books  WHERE rating >= 4          AND rating <= 5 AND total\_pages > 1300  ORDER BY      title; |

The output is as follows:



### 3) Using Db2 AND operator with parentheses

If you use the AND and OR operators with more than two conditions, you can use parentheses to explicitly specify the order in which you want Db2 to evaluate the conditions.

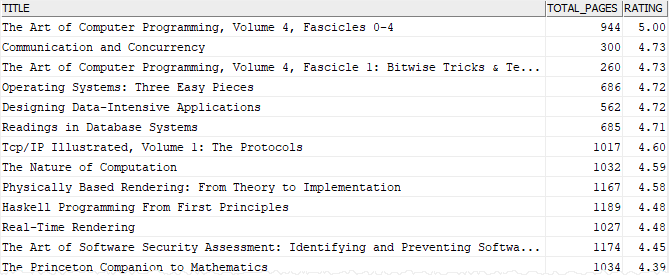
This example uses both AND and OR operators to find books that satisfy both the following conditions:

* The number of pages is greater than 1,000 and less than 1,200
* Rating is greater than 4.7

Query:



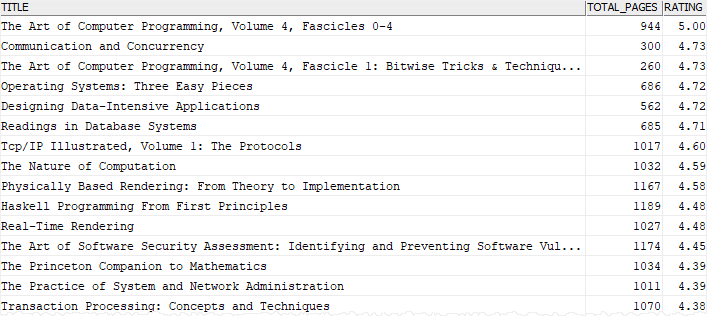
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      title,      total\_pages,      rating  FROM      books  WHERE(total\_pages < 1200          AND total\_pages > 1000)      OR rating > 4.7  ORDER BY      rating DESC; |



If you move the parentheses, the meaning of the [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause can change significantly:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      title,      total\_pages,      rating  FROM      books  WHERE total\_pages < 1200          AND (total\_pages > 1000      OR rating > 4.7)  ORDER BY      rating DESC; |



This query returns books that satisfy both the following conditions:

* The number of pages is less than 1200
* The number of pages is greater than 1000 or rating is greater than 4.7

### 4) Using Db2 NOT AND operator example

This example uses the NOT AND operator to find books that have the number of pages less than or equal 500 or rating greater than 4.7:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      title,      total\_pages,      rating  FROM      books  WHERE NOT(total\_pages > 500              AND rating < 4.7)  ORDER BY      total\_pages DESC,      rating DESC; |

# Db2 OR

**Summary**: in this tutorial, you will learn how to use the Db2 OR operator to query rows that satisfy either or both predicates of the search condition.

## Introduction Db2 OR operator

The OR operator is a logical operator that combines two Boolean expressions or predicates. the OR operator is often used in the [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause of the [SELECT](https://www.db2tutorial.com/db2-basics/db2-select/), [UPDATE](https://www.db2tutorial.com/db2-basics/db2-update/), and [DELETE](https://www.db2tutorial.com/db2-basics/db2-delete/) statements to specify a search condition for rows to be selected, updated, and deleted.

Here is the syntax of the OR operator:



|  |  |
| --- | --- |
| 1 | boolean\_expression1 OR boolean\_expression2 |

In this syntax, the boolean\_expression1 and boolean\_expression2 are the Boolean expressions that evaluate to true, false, and unknown.

The following table shows the result when combining true, false, and unknown values using the OR operator:

|  | **TRUE** | **FALSE** | **UNKNOWN** |
| --- | --- | --- | --- |
| **TRUE** | TRUE | TRUE | TRUE |
| **FALSE** | TRUE | FALSE | UNKNOWN |
| **UNKNOWN** | TRUE | UNKNOWN | UNKNOWN |

If you use both OR and [AND](https://www.db2tutorial.com/db2-basics/db2-and/) operators in an expression, Db2 always evaluates the AND operator first. To change the order of evaluation, you can use the parentheses.

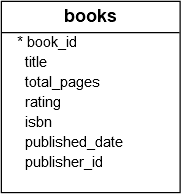
To negate the OR operator, you use the NOT operator as follows:



|  |  |
| --- | --- |
| 1 | NOT (boolean\_expression1 OR boolean\_expression2) |

## Db2 OR operator examples

Let’s use the books table from the [sample database](https://www.db2tutorial.com/getting-started/db2-sample-database/) to demonstrate the OR operator.



### 1) Db2 OR operator example

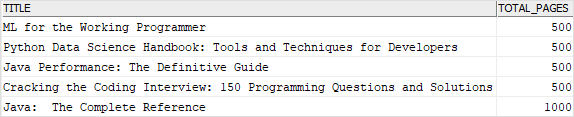
This example uses the OR operator to find the books that have the number of pages 500 or 1,000:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      title,      total\_pages  FROM      books  WHERE total\_pages = 500          OR total\_pages = 1000  ORDER BY      total\_pages; |

The query scans every row and returns the rows whose value in the total\_pages column is 500 or 1000.

Here is the result set:



Note that you can use the IN operator to achieve the same result:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      title,      total\_pages  FROM      books  WHERE      total\_pages IN (500, 1000)  ORDER BY      total\_pages; |

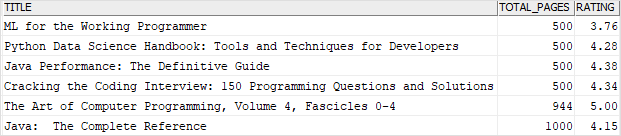
### 2) Using multiple Db2 OR operators example

This example uses two OR operators to find books whose ratings are 5 and the number of pages is 500 or 1000:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      title,      total\_pages,      rating  FROM      books  WHERE total\_pages = 500          OR total\_pages = 1000          OR rating = 5  ORDER BY      total\_pages; |

Here is the output:



### 3) Using Db2 OR operator with parentheses

If you use the OR and AND operators with more than two conditions, you can use parentheses to explicitly specify the evaluation order.

The following example uses both OR and AND operators to find books that satisfy either of the following conditions:

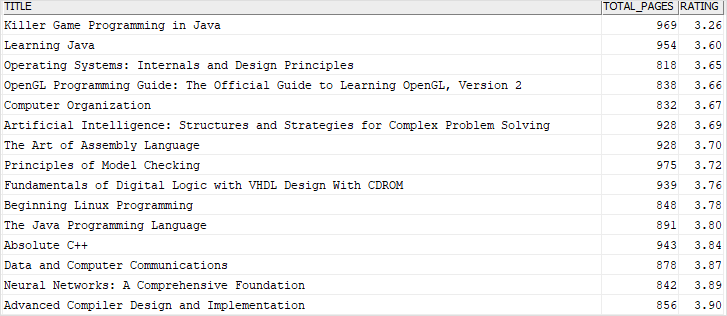
* The number of pages is greater than 800 and less than 1,000
* The rating is greater than 4.7

Here is the query:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | SELECT      title,      total\_pages,      rating  FROM      books  WHERE(total\_pages > 800          AND total\_pages < 1000)      OR rating > 4.7  ORDER BY      rating,      total\_pages; |

The following picture shows the partial output:



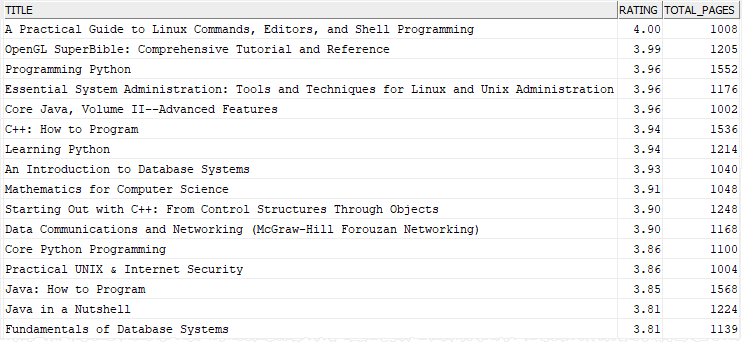
### 4) Using Db2 NOT OR operator example

This example uses the NOT OR operator to find books that have the number of pages greater than or equal 1000 and rating less than or equal 4:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | SELECT      title,      rating,      total\_pages  FROM      books  WHERE NOT (rating > 4 OR total\_pages < 1000)  ORDER BY      rating desc,      total\_pages desc; |

Here is the partial result set:



Note that based on the De Morgan’s laws, NOT (A OR B) is equivalent to (NOT A AND NOT B). In this case, the following conditions are equivalent:



|  |  |
| --- | --- |
| 1 | NOT (rating > 4 OR total\_pages < 1000) |

and



|  |  |
| --- | --- |
| 1 | rating <= 4 AND total\_pages >= 1000) |

# Db2 BETWEEN

**Summary**: in this tutorial, you will learn how to the Db2 BETWEEN operator to check whether a value lies between two other values.

## Introduction to Db2 BETWEEN operator

The Db2 BETWEEN operator is a logical operator that determines whether a value lies between two values that are specified in ascending order. The BETWEEN operator is often used in the [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause of the [SELECT](https://www.db2tutorial.com/db2-basics/db2-select/) statement to form the search condition for the rows returned by a query.

The following shows the syntax of the BETWEEN operator:



|  |  |
| --- | --- |
| 1 | expression BETWEEN low AND high |

In this syntax, low and high can be literal values or expressions. The BETWEEN operator returns true if the value of the expression lies between the low and high values, or simply:



|  |  |
| --- | --- |
| 1 | expression >= low AND expression <= high |

The NOT BETWEEN negates the BETWEEN operator. It returns true if the value is not between two other values:



|  |  |
| --- | --- |
| 1 | expression NOT BETWEEN low AND high |

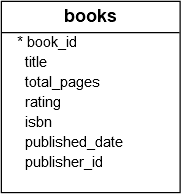
It is equivalent to the following expression:



|  |  |
| --- | --- |
| 1 | expression < low OR expression > high |

## Db2 BETWEEN operator examples

We will use the books table from the [sample database](https://www.db2tutorial.com/getting-started/db2-sample-database/) to demonstrate the BETWEEN operator:



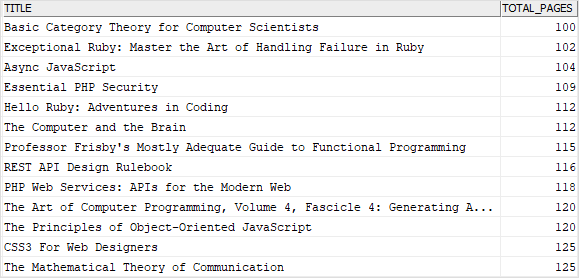
### 1) Using Db2 BETWEEN operator with numbers example

This example uses the BETWEEN operator to find books that have the number of pages between 100 and 125:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      title,      total\_pages  FROM      books  WHERE      total\_pages BETWEEN 100 AND 125  ORDER BY      total\_pages; |

Here is the result set:



It is equivalent to the following query:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      title,      total\_pages  FROM      books  WHERE      total\_pages >= 100 and total\_pages <= 125  ORDER BY      total\_pages; |

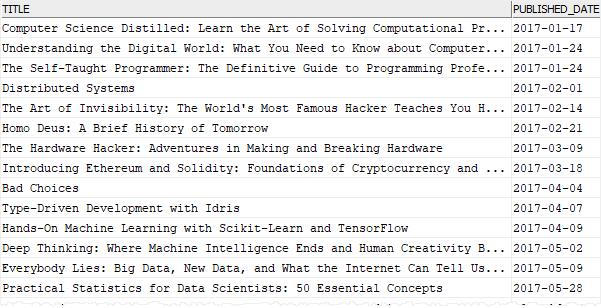
### 2) Using Db2 BETWEEN operator with dates example

The following example uses the BETWEEN operator finds books whose published dates are between Jan 01, 2017 and June 30, 2017:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      title,      published\_date  FROM      books  WHERE  published\_date BETWEEN '2017-01-01' AND '2017-06-30'  ORDER BY      published\_date; |

The following picture shows the output:

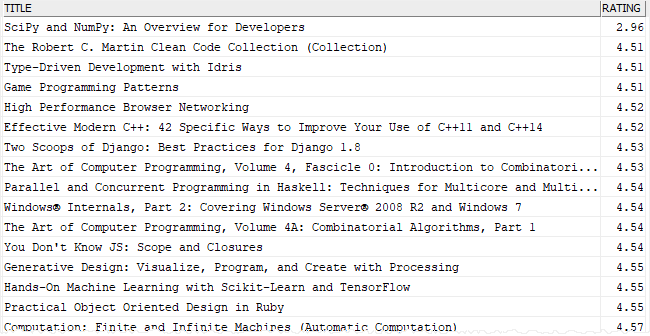


### 3) Db2 NOT BETWEEN operator example

This example uses the NOT BETWEEN operator to find books whose ratings are not between 3 and 4.5:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      title,      rating  FROM      books  WHERE rating NOT BETWEEN 3 AND 4.5  ORDER BY      rating; |



The query above is equivalent to the following query that uses comparison operators with the logical operator OR:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      title,      rating  FROM      books  WHERE rating < 3 OR rating > 4.5  ORDER BY      rating; |

# Db2 LIKE

**Summary**: in this tutorial, you will learn how to use Db2 LIKE operator to search for a string that has a certain pattern.

## Introduction to Db2 LIKE operator

The Db2 LIKE operator is a logical operator that returns true if a string contains a certain pattern. The pattern may include regular characters and special characters called wildcards.

The LIKE operator is used in the [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause of the [SELECT](https://www.db2tutorial.com/db2-basics/db2-select/), [UPDATE](https://www.db2tutorial.com/db2-basics/db2-update/), and [DELETE](https://www.db2tutorial.com/db2-basics/db2-delete/) statements to form the predicate for filtering rows based on a certain pattern.

The following shows the syntax of the LIKE operator:



|  |  |
| --- | --- |
| 1 | {column\_name | expression} LIKE pattern [ESCAPE escape\_character] |

### Pattern

The pattern is a string for which to search in the column\_name or expression. It may include the percent (%) and underscore (\_) wildcard characters.

* The percent wildcard (%) represents any string of zero or more characters.
* The underscore (\_) wildcard represents any single character.

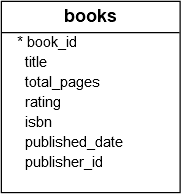
For example, s% matches any string starts with s and followed by any number of characters. %SQL% matches any string that contains the word SQL. S\_r matches any string that starts with S, followed by any character, and ended with the letter r like Sir, Ser, Str, Sor…

### ESCAPE escape\_character

Sometimes, you may want to find a pattern that contains the wildcard characters like % or \_. In this case, you can use the escape character. The escape character instructs the LIKE operator to treat the wildcard characters as the regular characters.

## Db2 LIKE operator examples

We will use the books table from the [sample database](https://www.db2tutorial.com/getting-started/db2-sample-database/) to demonstrate the LIKE operator.

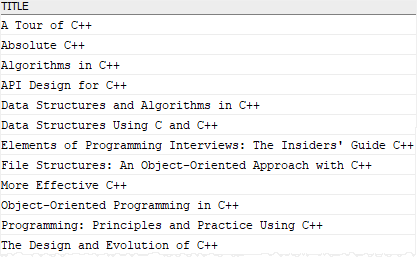


### 1) Using Db2 LIKE operator with % wildcard examples

This example uses the LIKE operator to find books that have the string C++ at the end of the title:



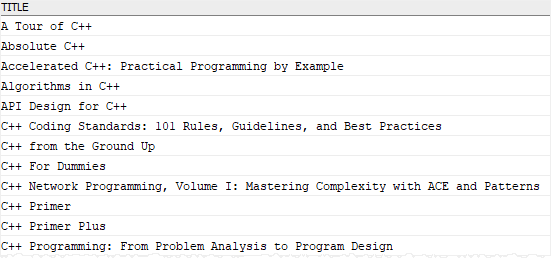
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      title  FROM      books  WHERE      title LIKE '%C++'  ORDER BY      title; |



To get books whose titles contain the string C++, you use the following query:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      title  FROM      books  WHERE      title LIKE '%C++%'  ORDER BY      title; |



### 2) Using LIKE operator with \_ wildcard examples

This example uses the LIKE operator to find books whose title starts with a pattern D\_t\_ e.g., Data, Database:

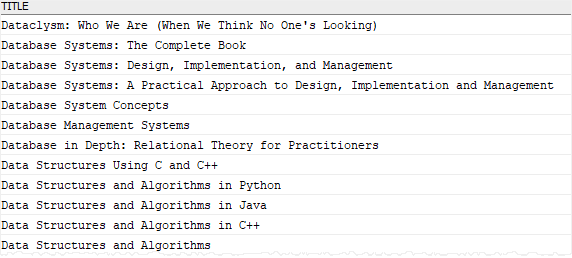


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      title  FROM      books  WHERE      title LIKE 'D\_t\_%'  ORDER BY      title desc; |

Let’s examine the pattern D\_t\_% in detail.

The matched string must start with the letter D, followed by any single character (\_), and then the letter t, and followed by any single character (\_), and finally followed by zero or more character (%).

Here is the output of the query:



## Db2 NOT LIKE operator

To negate the LIKE operator, you use the NOT operator as follows:



|  |  |
| --- | --- |
| 1 | {column\_name | expression} NOT LIKE pattern [ESCAPE escape\_character] |

This example uses the NOT LIKE operator to find books whose title does not contain the word Programming:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      title  FROM      books  WHERE      title NOT LIKE '%Programming%'  ORDER BY      title; |

# Db2 IN

**Summary**: in this tutorial, you will learn how to use the Db2 IN to compare a value with a set of values.

## Introduction to Db2 IN operator

The Db2 IN operator is a logical operator that compares a value with a set of values:



|  |  |
| --- | --- |
| 1 | expression IN (v1, v2, v3, ...) |

The IN operator returns true if the value of the expression matches one of the value in the list v1, v2, v3… Otherwise, it returns false.

The value list can be literal values as shown in the syntax above or a result set of a query. In this case, we called the query that supplies value to IN operator is a [subquery](https://www.db2tutorial.com/db2-basics/db2-subquery/). The subquery must return a single column or you will get an error.

The following illustrates the syntax of the IN operator with a subquery.



|  |  |
| --- | --- |
| 1 | expression IN (subquery) |

The IN operator is equivalent to multiple OR operators:



|  |  |
| --- | --- |
| 1  2  3  4 | expression = v1 OR  expression = v2 OR  expression = v3 OR  ... |

Using IN operator makes the code cleaner and easier to understand.

To negate the IN operator, you use the NOT operator as follows:



|  |  |
| --- | --- |
| 1 | expression NOT IN (v1, v2, v3, ...) |

It is equivalent to the following syntax:

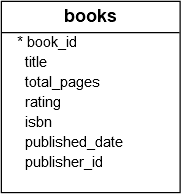


|  |  |
| --- | --- |
| 1  2  3  4 | expression <> v1 AND  expression <> v2 AND  expression <> v3 AND  ... |

You often use the IN operator in the [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause of the [SELECT](https://www.db2tutorial.com/db2-basics/db2-select/), DELETE, and UPDATE statements.

## Db2 IN operator examples

We will use the books table from the [sample database](https://www.db2tutorial.com/getting-started/db2-sample-database/) to demonstrates the IN operator.



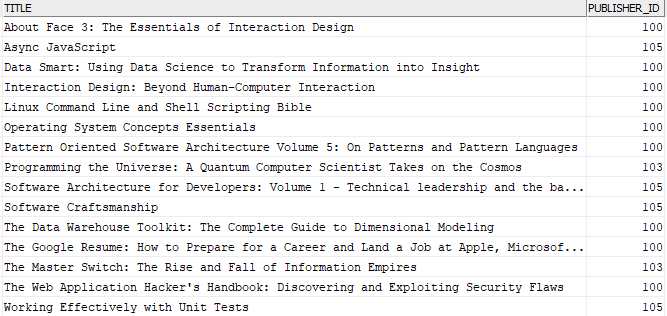
### 1) Using Db2 IN operator with a list of numeric values

This example uses the IN operator to find the books whose publisher id is in the list of 100, 103, and 105:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      title,      publisher\_id  FROM      books  WHERE      publisher\_id IN (100, 103, 105)  ORDER BY      title; |

Here is the result set:



The query above is equivalent to the following query that uses multiple OR operators:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      title,      publisher\_id  FROM      books  WHERE      publisher\_id = 100          OR publisher\_id = 103          OR publisher\_id = 105  ORDER BY      title; |

### 2) Using Db2 IN with a subquery

The following query uses the LIKE operator to find publisher ids of the publisher whose name starts with 'Addison Wesley':



|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT      publisher\_id  FROM      publishers  WHERE name LIKE 'Addison Wesley%'; |

It returns three rows:

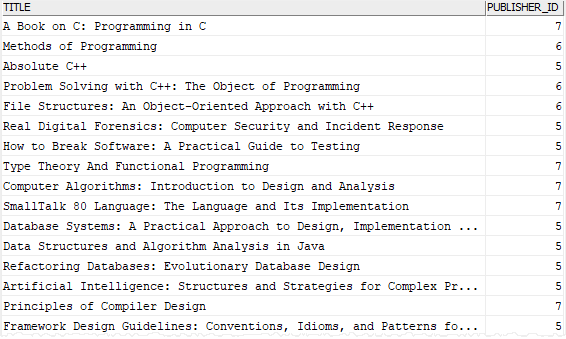


You can use this result set to supply to the IN operator to find books whose publisher name starts with 'Addison Wesley':



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | SELECT      title,      publisher\_id  FROM      books  WHERE publisher\_id IN  (      SELECT          publisher\_id      FROM          publishers      WHERE name LIKE 'Addison Wesley%'  ); |

Here is the partial output:



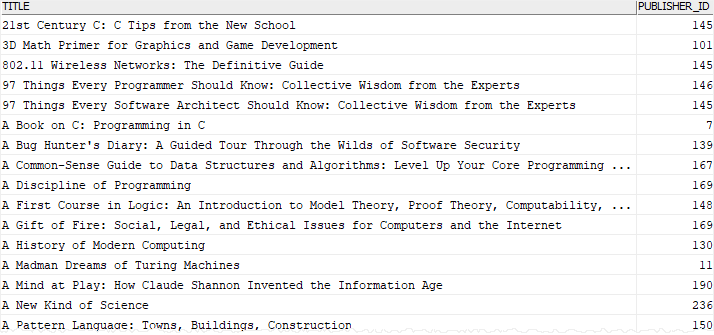
### 3) Using Db2 NOT IN operator example

The following example uses the NOT IN operator to find books whose publisher id is not in the list (100,103,105):



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      title,      publisher\_id  FROM      books  WHERE      publisher\_id NOT IN (100, 103, 105)  ORDER BY      title; |

The output is as follows:



It is equivalent to the following query that uses multiple AND operators:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | SELECT      title,      publisher\_id  FROM      books  WHERE publisher\_id <> 100          AND publisher\_id <> 103          AND publisher\_id <> 105  ORDER BY      title; |

# Db2 LIMIT

**Summary**: in this tutorial, you will learn how to use the db2 LIMIT clause to limit the number of rows returned by a query.

## Introduction to Db2 LIMIT clause

The LIMIT clause allows you to limit the number of rows returned by the query. The LIMIT clause is an extension of the SELECT statement that has the following syntax:



|  |  |
| --- | --- |
| 1  2  3  4 | SELECT select\_list  FROM table\_name  ORDER BY sort\_expression  LIMIT n [OFFSET m]; |

In this syntax:

* n is the number of rows to be returned.
* m is the number of rows to skip before returning the n rows.

Another shorter version of LIMIT clause is as follows:



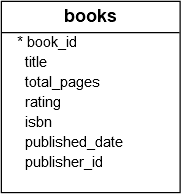
|  |  |
| --- | --- |
| 1 | LIMIT m, n; |

This syntax means skipping m rows and returning the next n rows from the result set.

A table may store rows in an unspecified order. If you don’t use the [ORDER BY](https://www.db2tutorial.com/db2-basics/db2-order-by/) clause with the LIMIT clause, the returned rows are also unspecified. Therefore, it is a good practice to always use the ORDER BY clause with the LIMIT clause.

## Db2 LIMIT clause examples

We will use the books table from the [sample database](https://www.db2tutorial.com/getting-started/db2-sample-database/) for the demonstration.

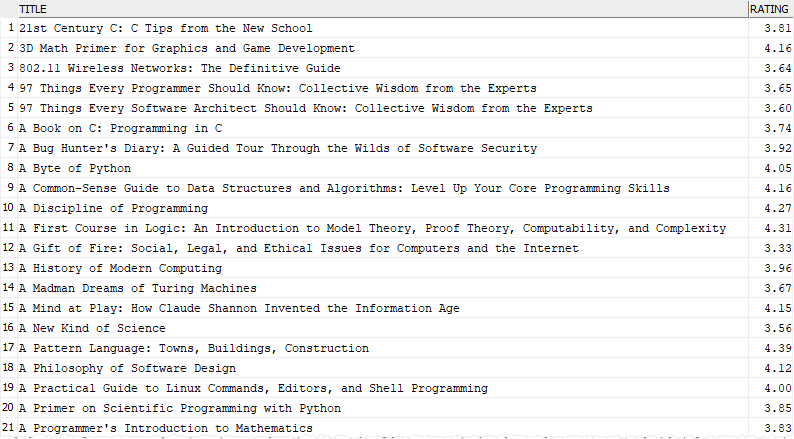


### 1) Db2 LIMIT OFFSET examples

The following query returns all rows from the books table sorted by titles:



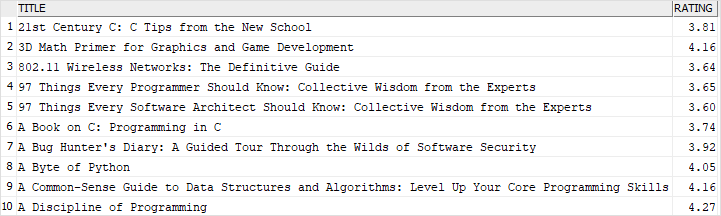
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT      title,      rating  FROM      books  ORDER BY      title; |



To get the first 10 books, you add the LIMIT clause to the above statement:



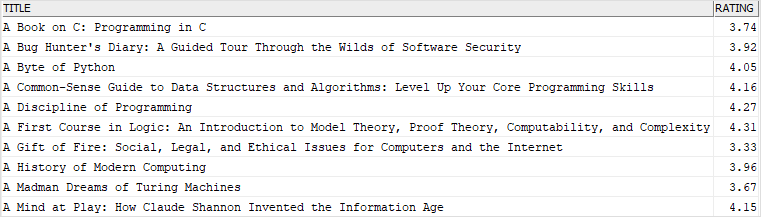
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      title,      rating  FROM      books  ORDER BY      title  LIMIT      10; |



To skip the first 5 books and return the next 10 books, you use the LIMIT OFFSET clause as follows:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      title,      rating  FROM      books  ORDER BY      title  LIMIT 10 OFFSET 5; |



Or in a shorter version:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      title,      rating  FROM      books  ORDER BY      title  LIMIT 5, 10; |

### 2) Using Db2 LIMIT to get top-N rows

The LIMIT clause is useful to get the top-N report e.g., top 10 books that have the highest rating and top 20 books that have the highest number of pages.

This example uses the LIMIT clause to get the top 10 books that have the highest ratings:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      title,      rating  FROM      books  ORDER BY      rating DESC  LIMIT 10; |

# Db2 FETCH

**Summary**: in this tutorial, you will learn how to use the Db2 FETCH clause to limit the number of rows returned by a query.

## Introduction to Db2 FETCH clause

When you use the [SELECT](https://www.db2tutorial.com/db2-basics/db2-select/) statement to query data from a table, you may get a large number of rows. However, you only need a small subset of these rows. Therefore, to limit the rows returned by a query, you use the FETCH clause as follows:



|  |  |
| --- | --- |
| 1  2 | OFFSET n ROWS  FETCH {FIRST | NEXT } m {ROW | ROWS} ONLY |

In this syntax:

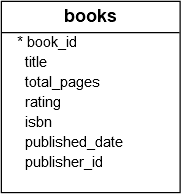
* n is the number of rows to skip.
* m is the number of rows to return. The FIRST and NEXT, ROW and ROWS are interchangeable respectively. They are used for the semantic purpose.

Notice that the FETCH clause is an ANSI-SQL version of the [LIMIT](https://www.db2tutorial.com/db2-basics/db2-limit/) clause.

Similar to the LIMIT clause, you should always use the FETCH clause with the ORDER BY clause to get the returned rows in a specified order.

## Db2 FETCH clause examples

We will use the books table from the [sample database](https://www.db2tutorial.com/db2-basics/db2-limit/) to demonstrate the FETCH clause.



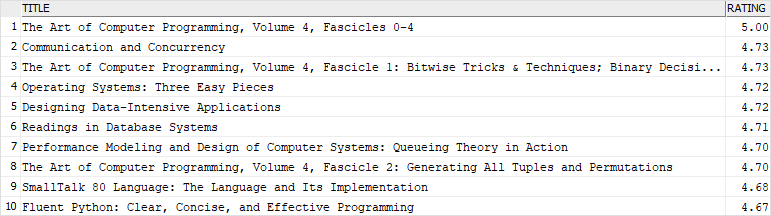
### 1) Using Db2 FETCH clause to get the top-N rows

This example uses the FETCH clause to get the top 10 books by ratings:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      title,      rating  FROM      books  ORDER BY      rating DESC  FETCH FIRST 10 ROWS ONLY; |

The following picture shows the output:



In this example:

* The ORDER BY clause sorts books by ratings from high to low.
* The FETCH clause picks only the first 10 rows, which have the highest ratings.

### 2) Using Db2 OFFSET FETCH for pagination example

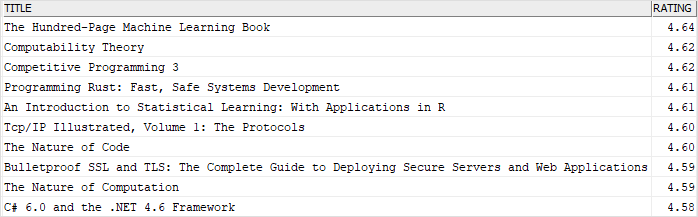
Suppose, you want to display books in pages sorted by ratings, each page has 10 books.

The following query uses the OFFSET FETCH clause to get the books on the second page:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      title,      rating  FROM      books  ORDER BY      rating DESC  OFFSET 10 ROWS  FETCH NEXT 10 ROWS ONLY; |

Here is the result set:



In this example:

* The OFFSET clause skips the first 10 rows which are on the first page.
* The FETCH clause picks the next 10 rows.

# Db2 Join

**Summary**: in this tutorial, you will learn how to query data from two tables using Db2 joins.

When you want to view the data from multiple tables, you can use the [SELECT](https://www.db2tutorial.com/db2-basics/db2-select/) statement with joins. The join associates the rows from one table with rows from another table based on a specified condition, typically of matching column values.

Db2 supports various kinds of joins including [inner join](https://www.db2tutorial.com/db2-basics/db2-inner-join/), [left outer join](https://www.db2tutorial.com/db2-basics/db2-left-join/), [right outer join](https://www.db2tutorial.com/db2-basics/db2-right-join/), and [full outer join](https://www.db2tutorial.com/db2-basics/db2-full-outer-join/).

Let’s setup some sample tables for demonstration.

## Setting sample tables

Second, create two new tables named contacts and customers:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | CREATE TABLE contacts (      contact\_id INT NOT NULL PRIMARY KEY,      name       VARCHAR(100) NOT NULL  );    CREATE TABLE customers (      customer\_id INT NOT NULL PRIMARY KEY,      name        VARCHAR(100) NOT NULL  ); |

Second, insert some rows into the contacts and customers tables:

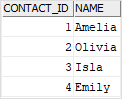


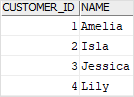
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | INSERT INTO contacts(contact\_id, name)  VALUES(1,'Amelia'),      (2,'Olivia'),      (3,'Isla'),      (4,'Emily');      INSERT INTO customers(customer\_id, name)  VALUES(1,'Amelia'),      (2,'Isla'),      (3,'Jessica'),      (4,'Lily'); |

Third, query data from the contacts and customers tables:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      \*  FROM      contacts;    SELECT      \*  FROM      customers; |





Let’s call the contacts table the left table and the customers table the right table.

## Db2 Inner Join

The inner join combines each row from the left table with rows of the right table, it keeps only the rows in which the join condition is true.

This example uses the [INNER JOIN](https://www.db2tutorial.com/db2-basics/db2-inner-join/) to get the rows from the contacts table that have the corresponding rows with the same values in the name column of the customers table:



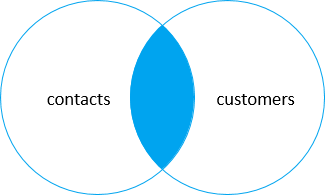
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      co.contact\_id,      co.name contact\_name,      cu.customer\_id,      cu.name customer\_name  FROM      contacts co      INNER JOIN customers cu          ON cu.name = co.name; |

Here is the result set:

db2 join inner join example

In this example, the join condition is cu.name = co.name which matches the values in the name column of the contacts table with the values in the name column of the customers table.

The following Venn diagram illustrates the result of the inner join of two result sets:



## Db2 Left Join

The [left join](https://www.db2tutorial.com/db2-basics/db2-left-join/) selects data starting from the left table and matches rows in the right table. Similar to the inner join, the left join returns all rows from the left table and the matching rows from the right table. In addition, if a row in the left table does not have a matching row in the right table, the columns of the right table will have nulls.

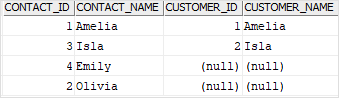
Note that the left join is also called the left outer join. The outer keyword is optional.

The following statement joins the contacts table with the customers table using left join:

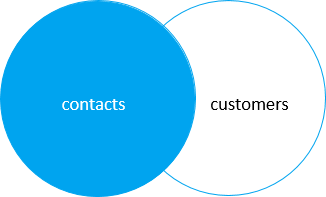


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      co.contact\_id,      co.name contact\_name,      cu.customer\_id,      cu.name customer\_name  FROM      contacts co      LEFT JOIN customers cu          ON cu.name = co.name; |

Here is the result set:



This Venn diagram illustrates the left join of two result sets:



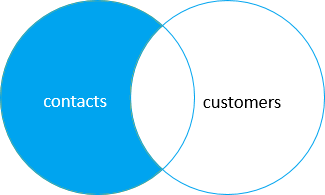
To get the rows that available only in the left table but not in the right table, you add a [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause to the above query:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      co.contact\_id,      co.name contact\_name,      cu.customer\_id,      cu.name customer\_name  FROM      contacts co      LEFT JOIN customers cu          ON cu.name = co.name  WHERE      cu.name IS NULL; |

db2 join left join with a where clause

And the this Venn diagram illustrates the left join that selects rows available only in the left table:



## Db2 Right Join

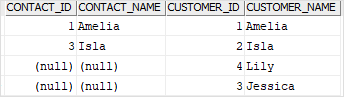
The [right join](https://www.db2tutorial.com/db2-basics/db2-right-join/) or [right outer join](https://www.db2tutorial.com/db2-basics/db2-right-join/), which is a reversed version of the left join, selects data starting from the right table and matches with the rows in the left table.

The right join returns a result set that includes all the rows from the right table and the matching rows in the left table. If a row in the right table does not have a matching row in the left table, all columns in the left table will contain nulls.

The following example uses the right join to query rows from contacts and customers tables:

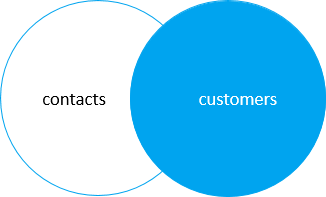


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      co.contact\_id,      co.name contact\_name,      cu.customer\_id,      cu.name customer\_name  FROM      contacts co      RIGHT JOIN customers cu          ON cu.name = co.name; |



Notice that all the rows from the right table (customers) are included in the result set.

Here is the Venn diagram of the right join:



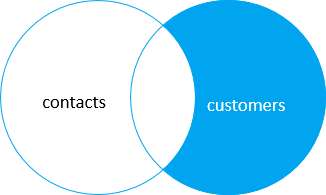
In order to get rows that are available only in the right table, you add a [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause to the above query:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      co.contact\_id,      co.name contact\_name,      cu.customer\_id,      cu.name customer\_name  FROM      contacts co      RIGHT JOIN customers cu          ON cu.name = co.name  WHERE      co.name IS NULL; |

db2 join right join with a where clause

And the following Venn diagram illustrates the operation:



## Db2 full join

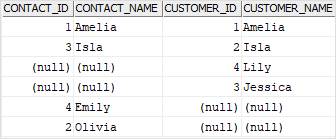
The [full join](https://www.db2tutorial.com/db2-basics/db2-full-outer-join/) returns a result set that includes all the rows from both left and right tables, with the matching rows from both sides where available. In case there is no match, the missing side will have nulls.

Note that full join and full outer join are synonyms. The outer keyword is optional.

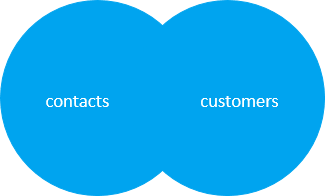
This example performs a full join between the contacts and customers tables:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      co.contact\_id,      co.name contact\_name,      cu.customer\_id,      cu.name customer\_name  FROM      contacts co      FULL JOIN customers cu          ON cu.name = co.name; |



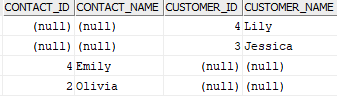
The Venn diagram that illustrates the full outer join:



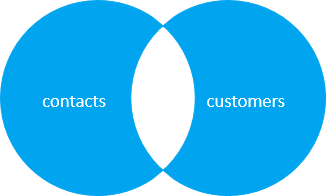
To select rows that are available in either left or right table, you exclude rows that are common to both tables by adding a [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause to the above query:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | SELECT      co.contact\_id,      co.name contact\_name,      cu.customer\_id,      cu.name customer\_name  FROM      contacts co      FULL OUTER JOIN customers cu          ON cu.name = co.name  WHERE      co.name IS NULL OR      cu.name IS NULL; |



The following Venn diagram illustrates the above operation:



# Db2 INNER JOIN

**Summary**: in this tutorial, you will learn how to use the Db2 INNER JOIN clause to query data from two or more related tables.

## Introduction to Db2 INNER JOIN clause

The INNER JOIN is one of the [join](https://www.db2tutorial.com/db2-basics/db2-join/) clauses that allow you to query data from two or more related tables. The INNER JOIN clause combines each row from the first table with every row from the second table, keeping only the rows in which the join condition evaluates to true.

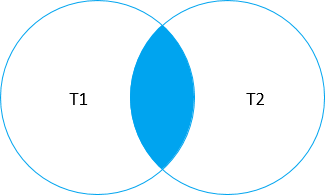
The following shows the syntax of joining two tables using the INNER JOIN clause:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT      select\_list  FROM      T1  INNER JOIN T2      ON join\_condition; |

In this syntax, the join\_condition is a Boolean expression that evaluates to true, false, and unknown. Typically, it matches the values of the columns in the table T1 with the values of the columns in the table T2 using the equality operator (=).

The following Venn diagram illustrates the inner join of two tables:



Note that if T1 and T2 tables have the same column names, you must fully qualify these column names in the query e.g., T1.id and T2.id or you will get an error. In case the table names are long, you can use the table aliases to save some typing.

To inner join more than two tables, you use multiple INNER JOIN clauses as shown in the following query:



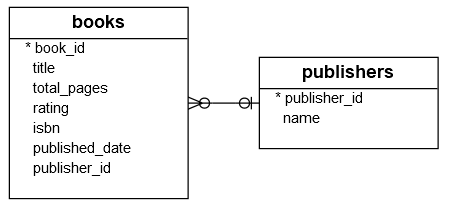
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT      select\_list  FROM      T1  INNER JOIN T2 ON join\_condition2  INNER JOIN T3 on join\_condition3  ...; |

## Db2 INNER JOIN examples

Let’s take some examples of using the INNER JOIN clause.

### 1) Using DB2 INNER JOIN to join two tables example

The following diagram shows the books and publishers tables:



In this model, one publisher may have zero or many books while each book belongs to zero or one publisher. The relationship between the books table and the publishers table is zero-to-many.

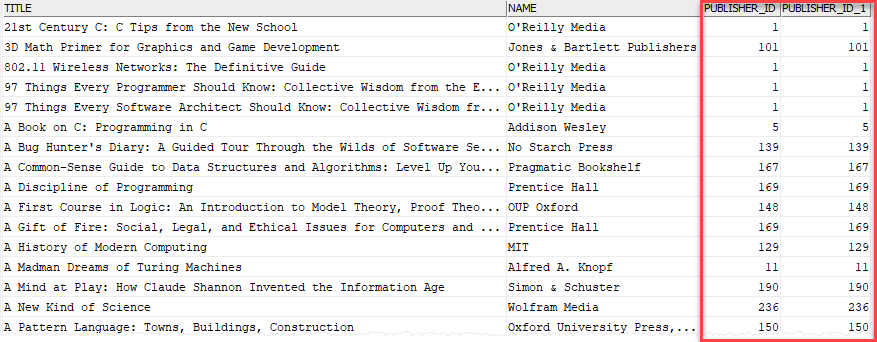
The publisher\_id column of the books table links to the publisher\_id column of the publishers table to establish this relationship.

The following example uses the INNER JOIN clause to join the books table with the publishers table:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      b.title,      p.name,      b.publisher\_id,      p.publisher\_id  FROM      books b  INNER JOIN publishers p      ON p.publisher\_id = b.publisher\_id  ORDER BY      b.title; |

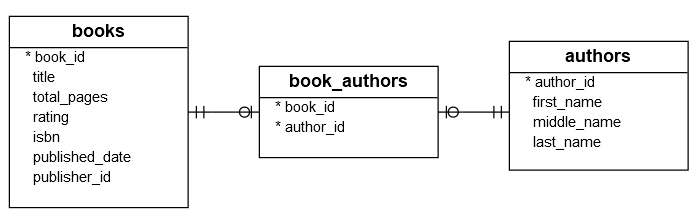
Here is the partial output:



In this example, the INNER JOIN clause compares the value in the publisher\_id column of each row in the books table with the value of the publisher\_id column of each row in the publishers table. If they are equal, The INNER JOIN combines columns of these two rows into a row and includes this row in the result set.

### 2) Using DB2 INNER JOIN to join three tables example

See the following books, authors, and book\_authors tables:



In this model, one book is written by one or many authors. And one author may write one or many books. The relationship between the books table and the authors table is many-to-many.

To model this many-to-many relationship, we have an associated table: book\_authors. Note that this associate table is also known as a junction table, a join table, or a cross-reference table.

In order to get the book titles from the books table and author’s names from the authors table, we join three tables using the INNER JOIN clause as follows:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | SELECT      b.title,      a.first\_name,      a.last\_name  FROM      books b  INNER JOIN book\_authors ba      ON ba.book\_id = b.book\_id  INNER JOIN authors a      ON a.author\_id = ba.author\_id  ORDER BY      b.title; |

# Db2 LEFT JOIN

**Summary**: in this tutorial, you will learn how to use the Db2 LEFT JOIN clause to query data from multiple tables.

## Introduction to Db2 LEFT JOIN clause

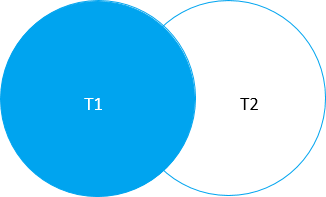
The LEFT JOIN clause is one of the [joins](https://www.db2tutorial.com/db2-basics/db2-join/) that allows you to [query data](https://www.db2tutorial.com/db2-basics/db2-select/) from two or more tables. Suppose, you have two tables: T1 and T2, called the left and the right tables respectively.

The LEFT JOIN clause selects data starting from the left table (T1). It compares each row in the left table with every row in the right table. If two rows match, the LEFT JOIN combines columns of these two rows into a row and includes this row in the final result set.

In case a row from the left table does not have any matching row from the right table, the LEFT JOIN also combines the columns of the row from the left table with the columns of the right table. However, the columns from the right table will have NULL values.

In other words, the LEFT JOIN clause returns all rows from the left table ( T1) and matching rows or NULL values from the right table ( T2).

This Venn diagram illustrates the LEFT JOIN of two tables T1 and T2:



The following shows the syntax of joining two tables using the LEFT JOIN clause:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT      select\_list  FROM      T1  LEFT JOIN T2      ON join\_condition; |

In this syntax, the join\_condition is a Boolean expression that evaluates to true, false, and unknown. Typically, it matches the values of the columns in the table T1 with the values of the columns in the table T2 using the equality operator (=).

Note that in case T1 and T2 tables have the same column names, you have to fully qualify these column names in the query e.g., T1.id and T2.id. If the table names are long, you can use the [table aliases](https://www.db2tutorial.com/db2-basics/db2-alias/) to save some typing.

To join more than two tables using the LEFT JOIN clause, you add more LEFT JOIN clauses as shown in the following query:



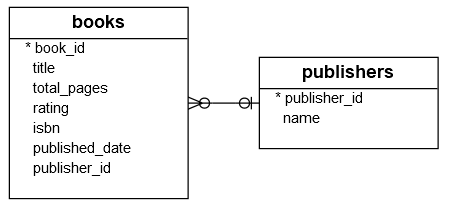
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT      select\_list  FROM      T1  LEFT JOIN T2 ON join\_condition2  LEFT JOIN T3 on join\_condition3  ...; |

## Db2 LEFT JOIN examples

Let’s take some examples of using the LEFT JOIN clause.

### 1) Using DB2 LEFT JOIN to join two tables example

The following diagram shows the books and publishers tables:



In this data model, a publisher may have zero or many books while each book belongs to zero or one publisher. The relationship between the books table and the publishers table is zero-to-many.

The publisher\_id column of the books table links to the publisher\_id column of the publishers table to establish this relationship.

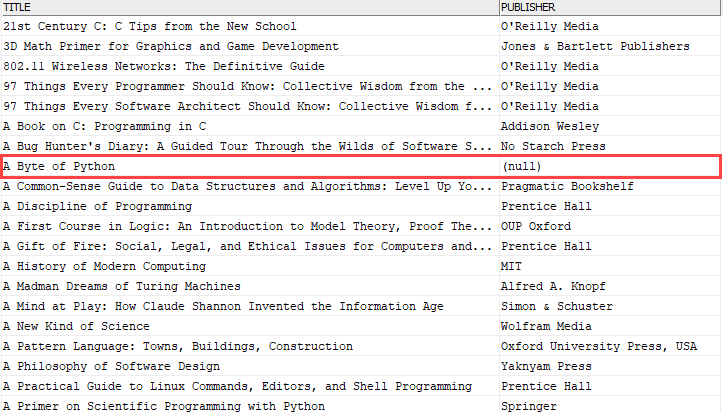
If a book does not associate with a publisher, maybe the publisher is unknown at the time of recording, the value in the publisher\_id column of the books table is NULL.

This query uses the LEFT JOIN clause to join the books table with the publishers table:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      b.title,      p.name  FROM      books b      LEFT JOIN publishers p          ON p.publisher\_id = b.publisher\_id  ORDER BY      b.title; |

The following picture shows the partial output:



In this example, the LEFT JOIN clause compares the value in the publisher\_id column of each row in the books table with the value in the publisher\_id column of each row in the publishers table. If they are equal, the LEFT JOIN combines columns of these two rows into a row and includes this row in the result set.

If a row from the books table that does not have a matching row in the publishers table, the publisher column in the result set will contain a NULL.

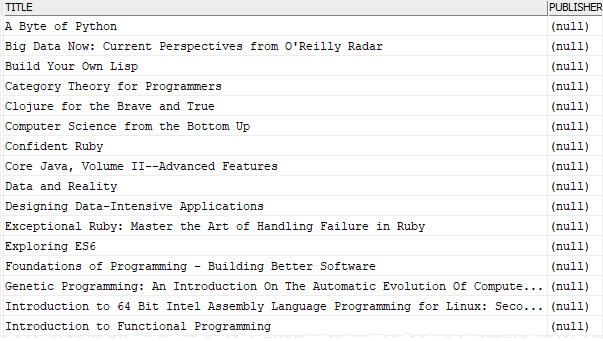
### 2) Using DB2 LEFT JOIN to find the missing rows in another table

The following query uses the LEFT JOIN clause to find books that do not have publishers:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      b.title,      p.name publisher  FROM      books b      LEFT JOIN publishers p          ON p.publisher\_id = b.publisher\_id  WHERE      p.name IS NULL  ORDER BY      b.title; |

Here is the partial result set:



# Db2 RIGHT JOIN

**Summary**: in this tutorial, you will learn how to use the Db2 RIGHT JOIN clause to query data from multiple tables.

## Introduction to Db2 RIGHT JOIN clause

The RIGHT JOIN clause is a reversed version of the [LEFT JOIN](https://www.db2tutorial.com/db2-basics/db2-left-join/) clause. The RIGHT JOIN clause allows you to query data from two or more tables.

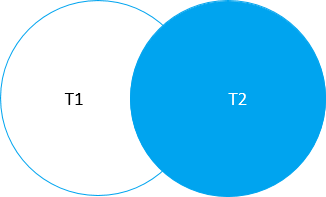
Suppose, you have two tables named T1 and T2, which are called the left table and the right table respectively.

The RIGHT JOIN clause selects data starting from the right table (T2). It compares each row in the right table (T2) with every row in the left table (T1). If two rows satisfy the join condition, the RIGHT JOIN clause combines columns of these rows into a new row and includes this new row in the result.

In case a row in the right table does not have a matching row in the left table, the RIGHT JOIN clause still combines the columns of the row in the right table with the columns of the row in the left table. However, the columns in the left table will have NULL values.

In other words, the RIGHT JOIN clause returns all rows from the right table (T2) and matching rows or NULL values from the left table (T1).

This Venn diagram illustrates how to use the RIGHT JOIN clause to join T1 and T2 tables:



The following shows the syntax of joining two tables using the RIGHT JOIN clause:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT      select\_list  FROM      T1  RIGHT JOIN T2      ON join\_condition; |

In this syntax, the join\_condition is a Boolean expression that evaluates to true, false, and unknown. Typically, it matches the values of the columns in the table T2 with the values of the columns in the table T1 using the equality operator (=).

Note that in case T1 and T2 tables have the same column names, you have to fully qualify these column names in the query like T1.id and T2.id. If the table names are long, you can use the table aliases to save some typing.

The following Venn diagram illustrates the RIGHT JOIN of two tables T1 and T2:

The following statement illustrates how to join more than two tables using the RIGHT JOIN clauses:



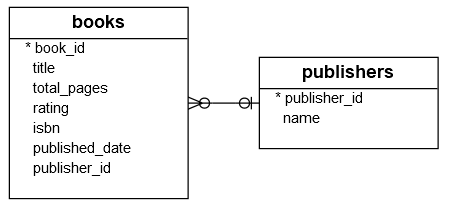
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT      select\_list  FROM      T1  RIGHT JOIN T2 ON join\_condition2  RIGHT JOIN T3 on join\_condition3  ...; |

## Db2 RIGHT JOIN examples

Let’s take some examples of using the RIGHT JOIN clause.

### 1) Using DB2 RIGHT JOIN to join two tables example

The following database diagram displays the books and publishers tables:



In this data model, a publisher may have zero or many books while a book may belong to zero or one publisher. The relationship between the books table and the publishers table is zero-to-many.

The publisher\_id column of the books table links to the publisher\_id column of the publishers table to establish this relationship.

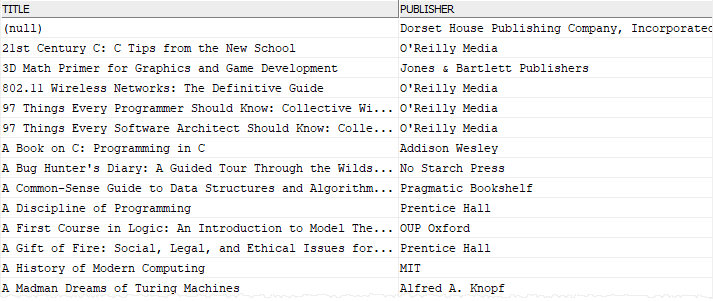
If a book does not associate with a publisher, maybe the publisher was not known at the time of recording, the value in the publisher\_id column of the books table is NULL.

This query uses the RIGHT JOIN clause to join the books table with the publishers table:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      b.title,      p.name  FROM      books b  RIGHT JOIN publishers p      ON p.publisher\_id = b.publisher\_id  ORDER BY      b.title NULLS FIRST; |

This picture shows the partial result set:



In this example, the RIGHT JOIN clause compares the value in the publisher\_id column of each row in the publishers table with the value of the publisher\_id column of each row in the books table. If they are equal, the RIGHT JOIN combines columns of these two rows into a row and includes this row in the result set.

If a row from the publishers table that does not have a matching row in the books table, the title column in the result set has a NULL.

### 2) Using DB2 RIGHT JOIN to find the missing rows in another table

The following query uses the RIGHT JOIN clause to find publishers that do not have books recorded:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      b.title,      p.name as publisher  FROM      books b  RIGHT JOIN publishers p      ON p.publisher\_id = b.publisher\_id  WHERE      title IS NULL  ORDER BY      b.title NULLS FIRST; |

Here is the result set:

https://www.db2tutorial.com/wp-content/uploads/2019/04/db2-right-join-find-missing-rows.png

# Db2 FULL OUTER JOIN

**Summary**: in this tutorial, you will learn how to use the Db2 FULL OUTER JOIN to query data from two tables.

## Introduction to Db2 FULL OUTER JOIN clause

Suppose you have two tables named T1 and T2, which are called the left table and the right table respectively.

The FULL OUTER JOIN of the T1 and T2 returns a result set which includes rows from both left and right tables. When no matching rows exist for the row from the left table, the columns of the right table are filled with NULL. Similarly, when no matching rows exist for the row from the right table, the columns of the left table will be filled with NULL.

The following shows the syntax of the FULL OUTER JOIN when joining two tables:



|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT      select\_list  FROM      T1  FULL OUTER JOIN T2 ON join\_condition; |

The OUTER keyword is optional so you can omit it as shown in the following query:

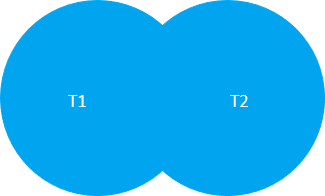


|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT      select\_list  FROM      T1  FULL JOIN T2 ON join\_condition; |

In this syntax:

* First, specify the left table T1 in the FROM clause.
* Second, specify the right table T2 and a join condition.

This Venn diagram illustrates the FULL OUTER JOIN of two result sets T1 and T2:

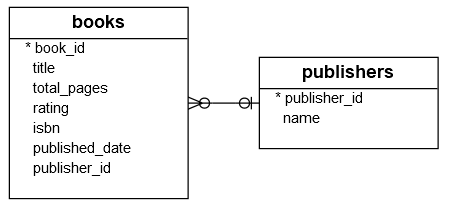


## Db2 FULL OUTER JOIN examples

Let’s take some examples of using the FULL OUTER JOIN clause.

### 1) Using Db2 FULL OUTER JOIN to join two tables example

This database diagram shows the books and publishers tables:



In this diagram, a publisher may have zero or many books while a book may belong to zero or one publisher. The relationship between the books table and the publishers table is zero-to-many. The publisher\_id column in both tables links a book to a publisher and vice versa.

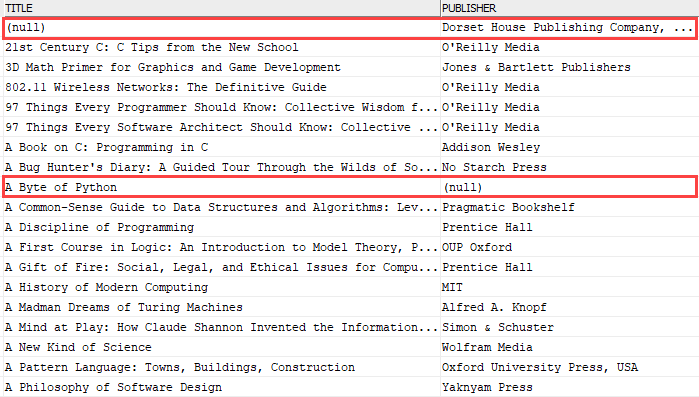
Note that in case a book does not associate with any publisher, maybe the publisher was unknown at the time of inserting, the value in the publisher\_id column of the books table is NULL.

The following query uses the FULL OUTER JOIN clause to join the books table with the publishers table:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | SELECT      b.title,      p.name AS publisher  FROM      books b      FULL OUTER JOIN publishers p          ON p.publisher\_id = b.publisher\_id  ORDER BY      b.title NULLS FIRST,      publisher; |

Here is the output:



In this example, the query returned books with publishers, books who do not have publishers, and publishers that do have any books.

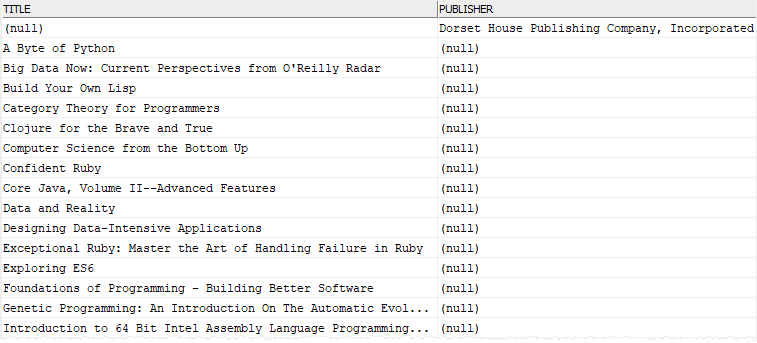
### 2) Using Db2 FULL OUTER JOIN to find the missing rows example

To find the publishers who do not have any books and books which do not associate with any publisher, you add a [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause to check if the book title [OR](https://www.db2tutorial.com/db2-basics/db2-or/) publisher name IS NULL:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | SELECT      b.title,      p.name AS publisher  FROM      books b      FULL OUTER JOIN publishers p          ON p.publisher\_id = b.publisher\_id  WHERE b.title IS NULL          OR p.name IS NULL  ORDER BY      b.title NULLS FIRST,      publisher; |

This picture shows the partial output:



# Db2 GROUP BY

**Summary**: in this tutorial, you will learn how to use the Db2 GROUP BY clause to group rows into groups.

## Introduction to Db2 GROUP BY clause

When you use the [SELECT](https://www.db2tutorial.com/db2-basics/db2-select/) statement to query data, you get a result set which consists of rows. To divide these rows into groups, you use the GROUP BY clause as shown in the following query:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT      select\_list  FROM      table\_name  GROUP BY      column1, column2,...; |

This statement divides rows derived from the FROM clause into groups by one or more column expression (column1, column2, …) specified in the GROUP BY clause.

When selecting groups of rows from the database, we are interested in the characteristics of the groups, not individual rows. Therefore, we often use [aggregate functions](https://www.db2tutorial.com/db2-aggregate-functions/) in conjunction with the GROUP BY clause.

An aggregate function takes multiple rows as an input and returns a single value for these rows. Some commonly used aggregate functions are [AVG()](https://www.db2tutorial.com/db2-aggregate-functions/db2-avg/), [COUNT()](https://www.db2tutorial.com/db2-aggregate-functions/db2-count/), [MIN()](https://www.db2tutorial.com/db2-aggregate-functions/db2-min/), [MAX()](https://www.db2tutorial.com/db2-aggregate-functions/db2-max/) and [SUM()](https://www.db2tutorial.com/db2-aggregate-functions/db2-sum/). For example, the COUNT() function returns the number of rows for each group. The AVG() function returns the average value of all values in the group.

Here is the common query that uses the GROUP BY clause with an aggregate function:

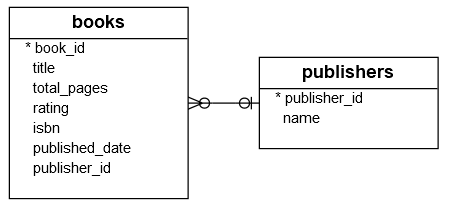


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | SELECT      column1,      column2,      ...      aggregate\_function(expression)    FROM      table\_name  GROUP BY      column1,      column2,      ...; |

Notice that any column listed in the select list that is not in the aggregate function expression must be placed in the GROUP BY clause, or you will get an error.

## Db2 GROUP BY clause examples

Let’s use the books and publishers tables from the [sample database](https://www.db2tutorial.com/getting-started/db2-sample-database/) for the demonstration.

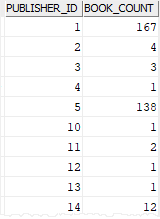


### 1) Using Db2 GROUP BY clause with COUNT(\*) function

This statement uses the GROUP BY clause with the [COUNT(\*)](https://www.db2tutorial.com/db2-aggregate-functions/db2-count/) function to find the number of books by publishers:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT      publisher\_id,      COUNT(\*) book\_count  FROM      books  GROUP BY      publisher\_id; |



In this statement:

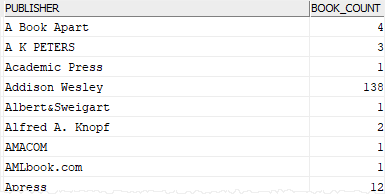
* The GROUP BY clause divides the rows in the books table into groups by the values in the publisher\_id column.
* The COUNT(\*) returns the number of rows per group.

If you want to get the publisher name instead of id, you can join the books table to the publishers table as shown in the following query:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      p.name  publisher,      COUNT(\*) book\_count  FROM      books b      INNER JOIN publishers p          ON p.publisher\_id = b.publisher\_id  GROUP BY      p.name  ORDER BY      publisher; |

Here is the output:

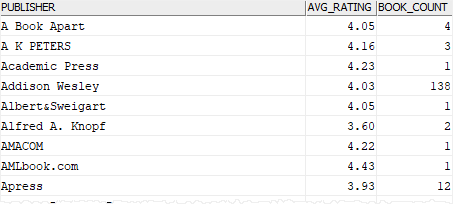


### 2) Using Db2 GROUP BY clause with AVG() function

This example uses the GROUP BY clause to find the average rating of all books for each publisher.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | SELECT      publishers.name publisher,      DECIMAL(AVG(rating),5,2) avg\_rating,      COUNT(\*) book\_count  FROM      books      INNER JOIN publishers          ON publishers.publisher\_id = books.publisher\_id  GROUP BY      publishers.name  ORDER BY      publisher; |



In this example:

* The GROUP BY clause divides the books into groups by the publisher names.
* The AVG() function returns the average rating of all books for every publisher.

Notice that we keep the COUNT(\*) function in the select list to view the number of books for each publisher.

### 3) Using Db2 GROUP BY clause with MIN() and MAX() functions

The following example finds the minimum & maximum ratings of books for each publisher using the [MIN()](https://www.db2tutorial.com/db2-aggregate-functions/db2-min/) and [MAX()](https://www.db2tutorial.com/db2-aggregate-functions/db2-max/) function with the GROUP BY clause:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | SELECT      publishers.name publisher,      MIN(rating) min\_rating,      MAX(rating) max\_rating  FROM      books      INNER JOIN publishers          ON publishers.publisher\_id = books.publisher\_id  GROUP BY      publishers.name  ORDER BY      publisher; |

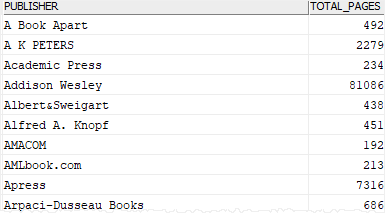


### 4) Using Db2 GROUP BY clause with SUM() function

This example uses the GROUP BY clause with the [SUM()](https://www.db2tutorial.com/db2-aggregate-functions/db2-sum/) function to find the total pages of all books for each publisher.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      publishers.name publisher,      SUM(total\_pages) total\_pages  FROM      books      INNER JOIN publishers          ON publishers.publisher\_id = books.publisher\_id  GROUP BY      publishers.name; |



# Db2 Subquery

**Summary**: in this tutorial, you will learn about the Db2 subquery or subselect which is a SELECT statement nested inside another statement such as SELECT, INSERT, UPDATE, and DELETE.

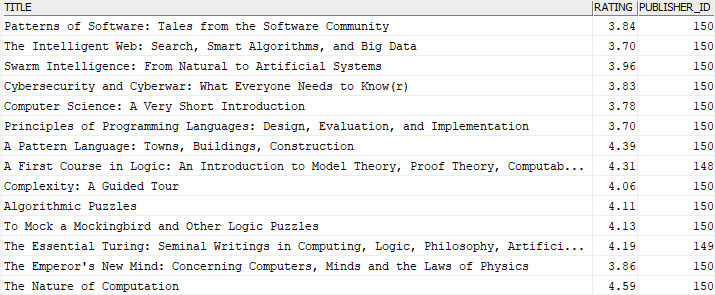
## Introduction to Db2 subquery

A subquery is a nested SQL statement that contains a SELECT statement inside the [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) or [HAVING](https://www.db2tutorial.com/db2-basics/db2-having/) clause of another SQL statement. A subquery is called a subselect.

The subquery allows you to form a search condition based on the data in another table. For example, you can find all books by publishers whose name contains the keyword Oxford:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | SELECT     title, rating, publisher\_id  FROM     books  WHERE     publisher\_id IN (        SELECT           publisher\_id        FROM           publishers        WHERE           name LIKE '%Oxford%'  ); |



In this example, here is the subquery:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT     publisher\_id  FROM     publishers  WHERE     name LIKE '%Oxford%'; |

The subquery is always enclosed in parentheses.

A subquery is also known as an inner query or inner select while the query that contains the subquery is known as an outer query or outer select.

To better understand the result of the outer query, you can imagine that Db2 goes through the following process:

1) DB2 first executes the subquery to get a list of publisher id:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT     publisher\_id  FROM     publishers  WHERE     name LIKE '%Oxford%'; |

Here is the output:



|  |  |
| --- | --- |
| 1  2  3  4  5 | PUBLISHER\_ID  -------------  148  149  150 |

2) DB2 then uses this list for the search condition of the outer query:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT     title,     rating,     publisher\_id  FROM     books  WHERE     publisher\_id IN (148,149,150); |

By using the subquery, you are able to combine steps together. The subquery removes the step of selecting the publisher id list and plugging them into the outer select. On top of it, the result of the query is automatically adjusted whenever the publisher data changes.

## Nested Subqueries

Db2 allows you to nest a subquery within another subquery. The relationship between the nested subquery and subquery is the same as the relationship between the subquery and outer query. The maximum level of nesting in Db2 is 15.

## Db2 Subquery types

Db2 allows you to use a subquery in the following:

* in the place of expression in the SELECT clause
* in the FROM clause
* within the IN or NOT IN operator in the WHERE clause.
* within the ANY or ALL operator in the WHERE clause
* within the EXISTS or NOT EXISTS operator in the WHERE clause.

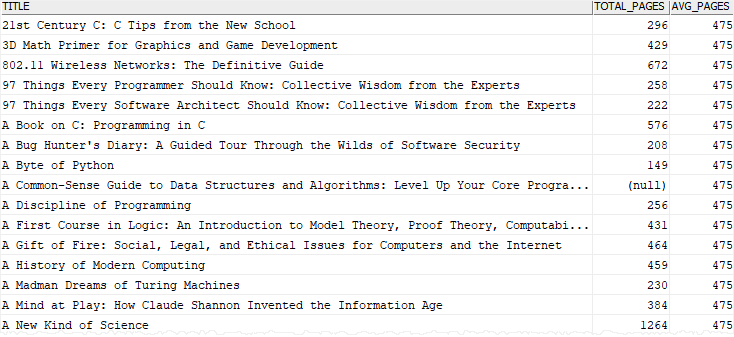
### 1) Using a subquery in place of an expression example

When a subquery returns a single value, you can place it in place of an expression e.g., in the select list of the SELECT clause.

This example uses a subquery to find the average number of pages of all books in the books table:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | SELECT     title,     total\_pages,     (SELECT        ROUND(AVG(total\_pages),0)     FROM        books     ) avg\_pages  FROM     books  ORDER BY     title; |



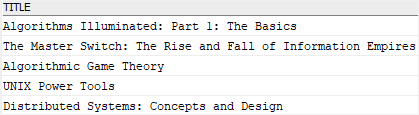
### 2) Using a subquery with IN operator example

You often use a subquery with the [IN](https://www.db2tutorial.com/db2-basics/db2-in/) operator. In this case, the subquery returns zero or multiple values. The outer query makes use of these values as a filter.

The following statement returns all books from the authors whose first name is Tim.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | SELECT     title  FROM     books b  INNER JOIN book\_authors A     ON A.book\_id = b.book\_id  WHERE author\_id IN (     SELECT        author\_id     FROM        authors     WHERE        first\_name = 'Tim'  ); |



In this example, the subquery returns a list of author ids which are used in the outer query to find their books.

### 3) Using subquery with ANY operator example

The following illustrates the syntax of a subquery used with the ANY operator:



|  |  |
| --- | --- |
| 1 | expression comparison\_operator ANY (subquery) |

Suppose, the subquery returns a list of value v1, v2, …. The ANY operator returns true if one of the following comparison pair evaluates to true:



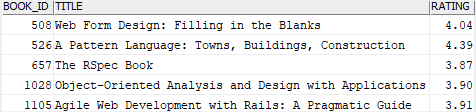
|  |  |
| --- | --- |
| 1  2  3 | (expression, v1)  (expression, v2)  (expression, ...) |

Note that the comparison operators are =, >, >=, <, <=, and <>.

The following statement uses a query in the ANY operator to find books which have more than five authors.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | SELECT     book\_id,     title,     rating  FROM     books  WHERE     book\_id = ANY(        SELECT           book\_id        FROM           book\_authors        GROUP BY           book\_id        HAVING           COUNT(author\_id) > 5     ); |



### 4) Using a subquery with ALL operator example

The syntax of using a subquery with the ALL operator is similar to the syntax of using the subquery with the ANY operator:



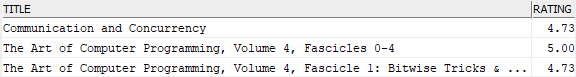
|  |  |
| --- | --- |
| 1 | expression comparison\_operator ALL (subquery) |

The ALL operator returns true if all comparison pairs return to TRUE.

This statement uses the ALL operator to find the books whose ratings are greater than the average rating of all books by publishers:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | SELECT     title,     rating  FROM     books  WHERE     rating > ALL(        SELECT           AVG(rating)        FROM           books        GROUP BY publisher\_id  ); |



### 5) Using a subquery with EXISTS operator example

Here is the syntax of using a subquery with EXISTS and NOT EXISTS operator:



|  |  |
| --- | --- |
| 1 | WHERE [NOT] EXISTS (subquery) |

The EXISTS operator returns true only if the subquery returns a non-empty result set. The NOT operator negates the EXISTS operator.

The following query finds the authors who have books published in 2019:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | SELECT     first\_name,     last\_name  FROM     authors a     INNER JOIN book\_authors b        ON b.author\_id = a.author\_id  WHERE     EXISTS (        SELECT           book\_id        FROM           books        WHERE           YEAR(published\_date) = 2018 AND             book\_id = b.book\_id  ); |



# Db2 HAVING

**Summary**: in this tutorial, you will learn how to use the Db2 HAVING clause to specify a search condition for groups.

## Introduction to Db2 HAVING clause

When you use the [SELECT](https://www.db2tutorial.com/db2-basics/db2-select/) statement to query data from one or more tables, you get a result set that contains all rows of the related tables. To specify a search condition for rows, you use the conditions in the [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause.

Similarly, to specify a search condition for the groups of rows returned by the [GROUP BY](https://www.db2tutorial.com/db2-basics/db2-group-by/) clause, you use the HAVING clause. The following illustrates the syntax of the HAVING clause:

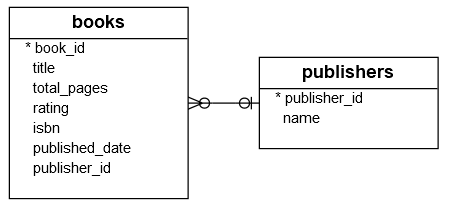


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | SELECT      select\_list  FROM      table\_name  GROUP BY      column1,      column2,      ...  HAVING      search\_condition; |

The search\_condition consists of one or more Boolean expressions that evaluate to true, false, or unknown. The statement returns only groups that satisfy the search\_condition. In other words, it returns only groups that cause the search\_condition to evaluate to true.

## Db2 HAVING clause examples

We will use the books and publishers tables from the [sample database](https://www.db2tutorial.com/getting-started/db2-sample-database/) to demonstrate the HAVING clause.

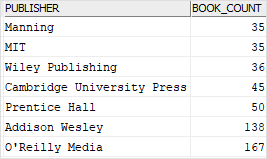


### 1) Using Db2 HAVING clause to filter groups example

This statement finds publishers that have more than 30 books:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | SELECT      p.name  publisher,      COUNT(\*) book\_count  FROM      books b      INNER JOIN publishers p          ON p.publisher\_id = b.publisher\_id  GROUP BY      p.name  HAVING      COUNT(\*) > 30  ORDER BY      book\_count; |



In this example:

* The GROUP BY and COUNT(\*) function returns publishers with their corresponding book counts
* The HAVING clause evaluates each group (publisher) and includes only the publishers that have more than 30 books.

### 2) Using Db2 HAVING clause to find duplicate rows

First, [create a new table](https://www.db2tutorial.com/db2-basics/db2-create-table/) named t1 for the demonstration.



|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE t1  (      id INT NOT NULL PRIMARY KEY,      c1 CHAR(1) NOT NULL  ); |

Second, [insert](https://www.db2tutorial.com/db2-basics/db2-insert/) some rows into the t1 table.



|  |  |
| --- | --- |
| 1  2 | INSERT INTO t1(id,c1)  VALUES(1,'A'),(2,'B'),(3,'C'),(4,'C'),(5,'A'); |

Third, [query](https://www.db2tutorial.com/db2-basics/db2-select/) data from the t1 table.



|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT      id,      c1  FROM      t1; |



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | ID          C1  ----------- --  1           A  2           B  3           C  4           C  5           A |

As you can see, the c1 column has some duplicate values e.g., A and C. Finding these duplicate rows in a table with many rows is not easy.

Fortunately, you can use the HAVING clause to find these duplicate values quickly:

* First, group the values in the column from which you want to find duplicates using the GROUP BY clause.
* Second, use the COUNT() function to get the number of values for each group.
* Third, use the HAVING clause to filter values whose the number of occurrences is greater than one.

Here is the query:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      c1,      COUNT(c1) value\_count  FROM      t1  GROUP BY      c1  HAVING      COUNT(c1) > 1; |

The following shows the output:



|  |  |
| --- | --- |
| 1  2  3  4 | C1 VALUE\_COUNT  -- -----------  A  2  C  2 |

# Db2 UNION

**Summary**: in this tutorial, you will learn how to use the Db2 UNION to combine the result sets of two or more subselects into a single result set.

## Db2 UNION Overview

Db2 UNION operator allows you to combine the result sets of two or more subselects into a single result set.

When Db2 encounters a UNION operator, it carries the following operations:

* First, process each subselect to form an interim result table.
* Then, combine these interim tables and delete the duplicate rows to form the final result set.

Here is the syntax of the Db2 UNION:



|  |  |
| --- | --- |
| 1  2  3  4 | subselect\_1  UNION  subselect\_2 |

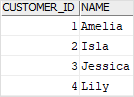
The queries in the above syntax must meet the following requirements:

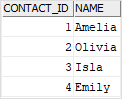
* Both subselects must have The number and the order of the columns.
* The data types of the corresponding columns must be the same or compatible via implicit conversion.

The UNION operator is useful to marge lists of values retrieved from multiple tables.

## Db2 UNION example

We will use the customers and contacts tables created in the [join tutorial](https://www.db2tutorial.com/db2-basics/db2-join/) for the demonstration.



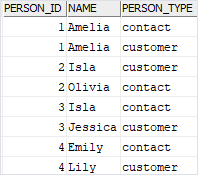


The following query uses the UNION operator to combine the customers and contacts into a single list. To distinguish customers and contacts, we add a new column named person\_type.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | SELECT     customer\_id person\_id,     name,     'customer' person\_type  FROM     customers  UNION  SELECT     contact\_id person\_id,     name,     'contact' person\_type  FROM     contacts; |

Here is the result set:



## Using UNION with ORDER BY clause

When you use the [ORDER BY](https://www.db2tutorial.com/db2-basics/db2-order-by/) clause in a query that uses the UNION operator:

1) You place the ORDER BY clause after the last subselect:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT     select\_list  FROM     table\_1  UNION  SELECT     select\_list  FROM     table\_2  ORDER BY     sort\_expression; |

2) You can use the column name in the ORDER BY clause if the column in the list has a name. In case the column of a result set does not have a name, you can use the column alias to assign it a name or use a positive integer in the ORDER BY clause to order the rows.



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT expression AS name  FROM table\_1  UNION  SELECT ...  FROM table\_2  ORDER BY name; |

or



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT expression  FROM table\_1  UNION  SELECT ...  FROM table\_2  ORDER BY 1 |

## DB2 UNION vs. UNION ALL

The UNION removes all duplicate rows by default. However, if you want to retain the duplicates, you use UNION ALL instead:



|  |  |
| --- | --- |
| 1  2  3 | subselect\_1  UNION ALL  subselect\_2 |

## Db2 UNION ALL example

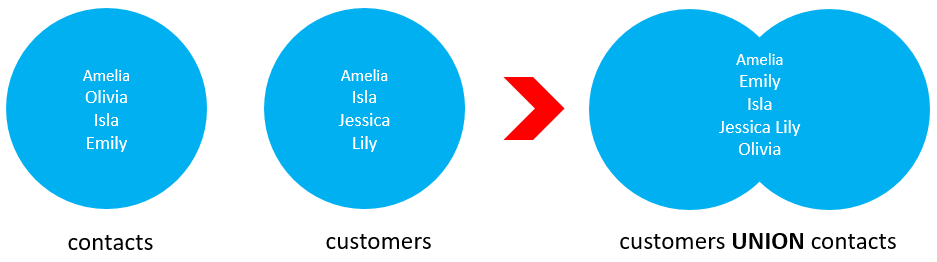
The following example uses UNION operator that removes all duplicate rows:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT     name  FROM     customers  UNION  SELECT     name  FROM     contacts  ORDER BY     name; |

Db2 UNION example

The following picture illustrates the union of two result sets: customers and contacts.



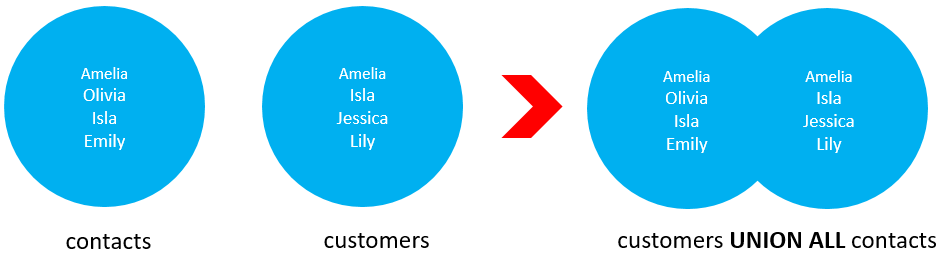
However, the following example uses the UNION ALL operator that retains the duplicate rows:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT     name  FROM     customers  UNION ALL  SELECT     name  FROM     contacts  ORDER BY     name; |

Db2 UNION ALL example

The following picture illustrates the union all of two result sets: customers and contacts.



## UNION vs. JOIN

The [join](https://www.db2tutorial.com/db2-basics/db2-join/) clause combines **columns** from two or more tables while the UNION operator combines **rows** from two or more subselects.

In other words, join appends the result sets from tables horizontally while UNION appends result sets from subselects vertically.

In this tutorial, you have learned how to use the Db2 UNION to combine rows from two or more subselects into a single result set.

# Db2 INTERSECT

**Summary**: in this tutorial, you will learn how to use the Db2 INTERSECT to combine two or more result sets and return only rows that exist in all result sets.

## Introduction to Db2 INTERSECT operator

The Db2 INTERSECT combines two or more result sets of subselects and returns only distinct rows that exist in all result sets.

Here is the syntax of the Db2 INTERSECT operator:

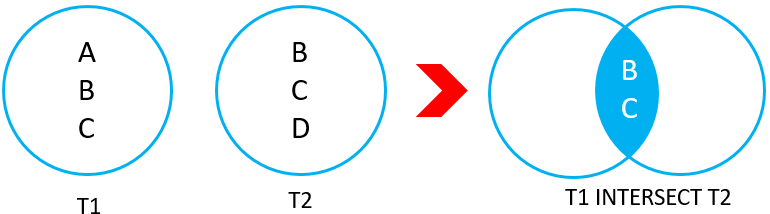


|  |  |
| --- | --- |
| 1  2  3 | subselect\_1  INTERSECT  subselect\_2 |

Like the [UNION](https://www.db2tutorial.com/db2-basics/db2-union/) operator, the subselects above must follow these rules:

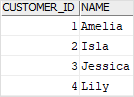
* The number and order of columns must be the same in all subselects.
* The data type of the columns (or expressions) in the select list of the subselects must be the same or at least compatible.

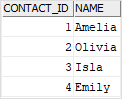
The following picture illustrates the INTERSECT operation of two result T1(A, B, C) and T2(B, C, D). The intersection of T1 and T2 result sets returns the distinct rows which are B and C:



## Db2 INTERSECT example

We will use the customers and contacts table created in the join tutorial for the demonstration:





This example uses the INTERSECT operator to find the contacts who are also the customers:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT     name  FROM     customers  INTERSECT  SELECT     name  FROM     contacts; |

Here is the output:

Db2 INTERSECT example

# Db2 EXCEPT

**Summary**: in this tutorial, you will learn how to use the Db2 EXCEPT to return the result of the first subselect minus any matching rows from the result set of the second subselect.

## Db2 EXCEPT Operator Overview

The Db2 EXCEPT combines two or more result sets of subselects. It returns the result set of the first subselect minus any matching rows of the result of the second subselect.

Here is the syntax of the Db2 EXCEPT operator:

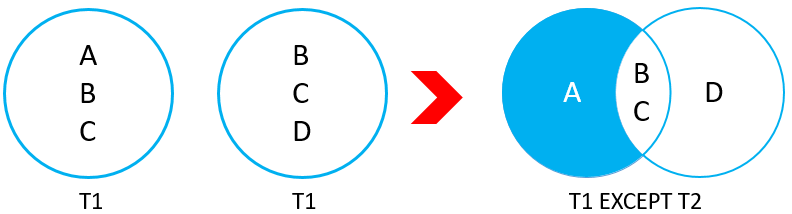


|  |  |
| --- | --- |
| 1  2  3 | subselect\_1  EXCEPT  subselect\_2 |

The columns and expression in the select list of the subselects must follow these rules:

* The number and order of columns or expressions must be the same in both subselects.
* The data types of the corresponding columns or expressions must be the same or compatible.

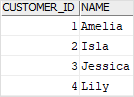
The following picture illustrates the EXCEPT operation of the two result sets T1 (A, B, C) and T2 (B, C, D):

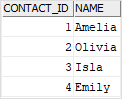


The except of T1 and T2 returns A which is the distinct rows from the T1 result set that does not appear in the T2 result set.

## Db2 EXCEPT example

We’ll use the customers and contacts tables created in the [join tutorial](https://www.db2tutorial.com/db2-basics/db2-join/) for the demonstration:





The following example uses the EXCEPT operator to find the customers who are not in the contacts table:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT     name  FROM     customers  EXCEPT  SELECT     name  FROM     contacts; |

Here is the output:



|  |  |
| --- | --- |
| 1  2  3  4 | NAME  -----------  Jessica  Lily |

# Db2 INSERT Multiple Rows

**Summary**: in this tutorial, you will learn how to insert multiple rows into a table by using a single INSERT statement.

## Db2 INSERT multiple rows statement overview

The Db2 INSERT statement allows you to insert multiple rows into a table using the following syntax:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | INSERT INTO      table\_name(column\_list)  VALUES      (value\_list\_1),      (value\_list\_2),      (value\_list\_3),      ...; |

To insert multiple rows into a table, you need to:

* First, specify the name of the table and a list of columns in parentheses.
* Second, use a list of comma-separated lists of column values. Each item in the list represents a row that will be inserted into the table.

## DB2 INSERT multiple rows example

We will use the lists table created in the [insert tutorial](https://www.db2tutorial.com/db2-basics/db2-insert/).

### 1) Inserting multiple rows into a table example

The following statement inserts three rows into the lists table:



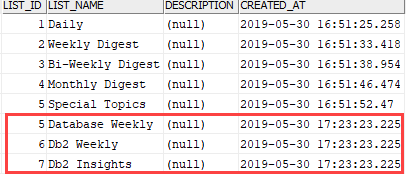
|  |  |
| --- | --- |
| 1  2  3  4  5 | INSERT INTO lists(list\_name)  VALUES  ('Database Weekly'),  ('Db2 Weekly'),  ('Db2 Insights'); |

To verify the data in the lists table after inserting, you can use the following query:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM lists; |

Here is the output:



### 2) Inserting multiple rows into a table and return a list of ids

To insert multiple rows into the lists table and return a list of inserted ids, you use the following statement:

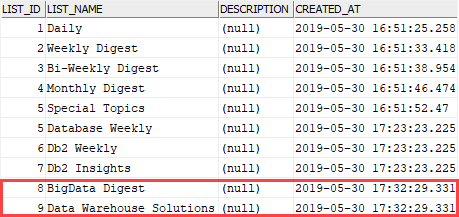


|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      list\_id  FROM FINAL TABLE (      INSERT INTO lists(list\_name)      VALUES          ('BigData Digest'),          ('Data Warehouse Solutions')  ); |

The statement returns the following output:

Db2 Insert Multiple Rows and return id list example

The following SELECT statement verifies the data of the lists table after insert:



# Db2 UPDATE

**Summary**: in this tutorial, you will learn how to use the Db2 UPDATE statement to modify data in a table.

## Db2 UPDATE statement overview

To change the existing data in a table, you use the following UPDATE statement. Here is its syntax:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | UPDATE     table\_name  SET     c1 = v1,     c2 = v2,     ... ,     cn = vn  [WHERE condition] |

In this syntax:

* First, specify the name of the table that you want to update data.
* Second, specify a list of column c1, c2, …, cn and the corresponding value v1, v2, … vn that need to be updated.
* Third, specify the condition to indicate which rows to be updated. Any row that causes the condition in the [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause to evaluate to true will be updated. The WHERE clause is optional, if you omit it, the UPDATE statement will update all rows in the table.

You can also use the following syntax of the UPDATE statement to update the data in a table:

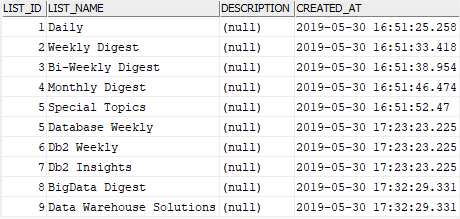


|  |  |
| --- | --- |
| 1  2  3  4  5 | UPDATE     table\_name  SET     (c1, c2, ... cn) = (v1, v2..., vn)  WHERE condition |

## Db2 UPDATE examples

We’ll use the lists table created in the [INSERT](https://www.db2tutorial.com/db2-basics/db2-insert/) statement tutorial.

Here is the data of the lists table:



### 1) Using Db2 UPDATE to update a single row example

The following example uses the UPDATE statement to update a single row in the lists table:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | UPDATE     lists  SET     description = 'General topics that sent out daily'  WHERE     list\_id = 1; |

In this example, we used a condition in the WHERE clause that specifies the row whose list\_id is 1 to be updated.

Db2 issued the following message:



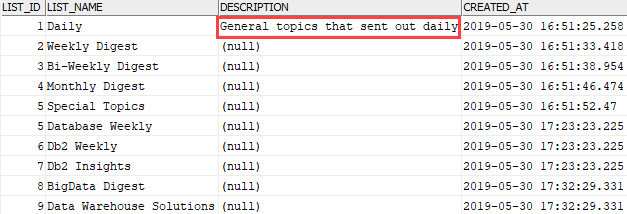
|  |  |
| --- | --- |
| 1 | (1 rows affected) |

After executing the statement, you can view the modified data in the lists table by using the following SELECT statement:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT     list\_id,     list\_name,     description,     created\_at  FROM     lists; |

As you can see, the value in the description has been updated successfully.



### 2) Using Db2 UPDATE to update multiple rows example

The following statement updates the description of rows whose description is NULL to 'N/A':



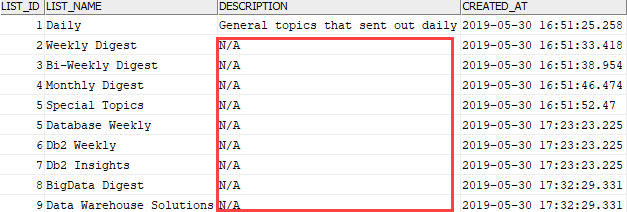
|  |  |
| --- | --- |
| 1  2  3  4  5  6 | UPDATE     lists  SET     description = 'N/A'  WHERE     description IS NULL; |

Here is the output:



|  |  |
| --- | --- |
| 1 | 9 rows updated. |

It means that 9 rows have been updated.



# Db2 DELETE

**Summary**: in this tutorial, you will learn how to delete one or more rows in a table using the Db2 DELETE statement.

## Introduction to Db2 DELETE statement

The DELETE statement allows you to delete one or more rows from a table. The following illustrates the syntax of the DELETE statement:



|  |  |
| --- | --- |
| 1  2 | DELETE FROM table\_name  [WHERE condition]; |

In this syntax:

* First, specify the name of the table from which you want to delete data.
* Second, use a condition in the [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause to specify which rows to delete. All rows that cause the condition to evaluate to true will be deleted.

The WHERE clause is optional. If you skip it, the DELETE statement will remove all rows from the target table.

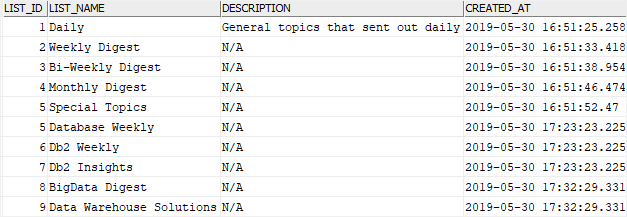
If no row satisfies the condition, the DELETE statement just does nothing.

Typically, a table is associated with another table via a relationship: one-to-one, one-to-many, or many-to-many. Depending on the setting of the [foreign key constraint](https://www.db2tutorial.com/db2-basics/db2-foreign-key/) such as ON DELETE CASCADE, the DELETE statement will automatically delete the rows from the child table when a row from the parent table is deleted.

## Db2 DELETE statement examples

We will use the lists table created in the [INSERT](https://www.db2tutorial.com/db2-basics/db2-insert/) statement for the demonstration.

This picture displays the data from the lists table:



### 1) Deleting one row from a table example

The following DELETE statement removes the row with id 1 from the lists table:



|  |  |
| --- | --- |
| 1  2 | DELETE FROM lists  WHERE list\_id = 1; |

Here is the message indicating that one row has been deleted:

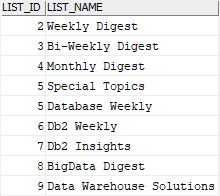


|  |  |
| --- | --- |
| 1 | 1 rows affected |

You can verify the deletion by using the following SELECT statement:



|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT     list\_id,     list\_name  FROM     lists; |



### 2) Deleting multiple rows from a table example

The following example uses a condition in the [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause to delete multiple rows from the lists table.



|  |  |
| --- | --- |
| 1  2 | DELETE FROM lists  WHERE list\_id IN (2,3,4); |

Db2 issued the following message indicating that 3 rows have been deleted successfully.



|  |  |
| --- | --- |
| 1 | 3 rows deleted. |

### 3) Deleting all rows from a table example

The following example uses the DELETE statement to remove all rows from the lists table:



|  |  |
| --- | --- |
| 1 | DELETE FROM lists; |

The output is as follows:



|  |  |
| --- | --- |
| 1 | 6 rows deleted |

# Db2 CREATE TABLE

**Summary**: in this tutorial, you will learn how to create a new table in the database by using the Db2 CREATE TABLE statement.

## Introduction to the Db2 CREATE TABLE statement

Tables are logical objects in Db2 that stores data. Tables are made up of columns and rows, like a spreadsheet. The rows of a table do not have a specified order. However, the columns of the table have the order that is specified when the table created.

A column in a table is associated with a specific data type e.g., character string, numeric, or temporal. A column always stores values of the same type. A table can have one or more columns. Typically, a table contains multiple columns.

To create a new table, you use the CREATE TABLE statement. The following shows the simplified syntax of the CREATE TABLE statement:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE TABLE [schema\_name.]table\_name (     column\_1 data\_type NOT NULL,     column\_2 data\_type DEFAULT value,     column\_3 data\_type CHECK(expression),     ...,     table\_constraints  ); |

In this syntax:

* First, specify the name of the table that you want to create. The name of the table must be unique within the schema. You can optionally specify the schema to which the table belongs.
* Second, specify a list of columns of the table. Each column is associated with a specific data type and may have constraint such as [NOT NULL](https://www.db2tutorial.com/db2-basics/db2-not-null/)and [CHECK](https://www.db2tutorial.com/db2-basics/db2-check-constraint/) constraint.
* Third, specify the table constraint such as [PRIMARY KEY](https://www.db2tutorial.com/db2-basics/db2-primary-key/), [FOREIGN KEY](https://www.db2tutorial.com/db2-basics/db2-foreign-key/) and [CHECK](https://www.db2tutorial.com/db2-basics/db2-check-constraint/) constraints. Note that you’ll learn more about these constraints in the next tutorial.

## Db2 CREATE TABLE example

The following example uses the CREATE TABLE statement to create a new table named stores:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | CREATE TABLE stores(     store\_id INT GENERATED BY DEFAULT AS IDENTITY NOT NULL,     store\_name VARCHAR(150) NOT NULL,     address\_line\_1 VARCHAR(255) NOT NULL,     address\_line\_2 VARCHAR(100),     city\_id INT NOT NULL,     state\_id INT NOT NULL,     zip\_code VARCHAR(6),     PRIMARY KEY (store\_id)  ); |

In this stores table:

* The store\_id column is an integer column. The GENERATED BY DEFAULT AS IDENTITY clause marks the store\_id column as an identity column so that when you [insert a new row](https://www.db2tutorial.com/db2-basics/db2-insert/) into the stores table, Db2 will automatically generate a sequential integer for the store\_id column. The NOT NULL constraint ensures that the store\_id will not accept any NULL value.
* The store\_name is a varying character (VARCHAR) column with a maximum length of 150. It has a NOT NULL constraint that will enforce non-null values.
* The address\_line\_1 is also a varying character column with a maximum length of 255 and does not accept NULL.
* The address\_line\_2 is a varying character column with a maximum length of 100. The address\_line\_2 is a nullable column so it can store NULL values.
* The city\_id and state\_id are integer columns. They accept NULL values.
* The zip\_code column is a varying character column with a maximum length of 6. It is a nullable column.
* The store\_id is the primary key column of the stores table specified by the PRIMARY KEY constraint at the end of the statement. It means that the store\_id will store unique values that identify all rows of the table.

After executing the CREATE TABLE statement, you will find the stores table appears in the database catalog. The stores table is empty once created. You can view its content by using the following SELECT statement:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM stores; |

To add new rows to the stores table, you use the [INSERT](https://www.db2tutorial.com/db2-basics/db2-insert/) statement.

# Db2 Identity Column

**Summary**: in this tutorial, you will learn how to use DB2 identity column to define an identity column for a table.

## Introduction to Db2 identity column

When you [create a new table](https://www.db2tutorial.com/db2-basics/db2-create-table/) and use the GENERATED AS IDENTITY option for a column, this column will become an identity column.

An identity column contains a unique integer for each row in the table. When you [insert a new row](https://www.db2tutorial.com/db2-basics/db2-insert/) into the table, Db2 automatically generates a sequential integer for the identity column. Thus, identity columns are ideal for the primary key columns such as book id (book\_id) or publisher id (publisher\_id).

The following shows the syntax of declaring an identity column:



|  |  |
| --- | --- |
| 1  2  3 | column\_name data\_type     GENERATED { ALWAYS | BY DEFAULT } AS IDENTITY     [( identity\_option) ] |

In this syntax:

First, specify the data type for the identity column. The data type can be SMALLINT, INT, and BIGINT.

Second, use either GENERATED ALWAYS or GENERATED BY DEFAULT option.

* For the GENERATED ALWAYS option, Db2 will always generate a sequential integer for the identity column. Any attempt to insert a value into the identity column with GENERATED ALWAYS option will result in an error.
* On the other hand, for the GENERATED BY DEFAULT option, DB2 will only generate the sequential integer when you don’t provide the value for the identity column. If you [insert a value](https://www.db2tutorial.com/db2-basics/db2-insert/) into the identity column with the GENERATED BY DEFAULT option, Db2 will use your value instead of using the system generated one.

Third, specify the identity column’s options:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | (     START WITH starting\_value     INCREMENT BY increment\_value     [MINVALUE min\_value]     [MAXVALUE max\_value]     [CYCLE | NO CYCLE]  ) |

The identity option allows you to specify the starting value in START WITH clause and increment value in the INCREMENT BY.

If the increment value is positive, you will have an ascending sequence like 1, 2, 3, … In case it is negative, then you will have a descending sequence e.g., -1, -2, -3, …

The MINVALUE and MAXVALUE options allow you to specify the minimum and maximum values that Db2 will generate.

The CYCLE or NOCYCLE option determines whether Db2 should restart the values when it has generated all the possible values.

For example, if you use CYCLE option and the sequence is 1, 2, 3, then Db2 will return 1 if it has generated 3. However, if you use the NO CYCLE option, Db2 will raise an error instead.

Notice that a table can have one and only one identity column in a table in Db2.

## Db2 identity column examples

Let’s take some examples of using identity columns to get a better understanding.

### 1) Db2 identity column example

First, [create a new table](https://www.db2tutorial.com/db2-basics/db2-create-table/) named t1 with the id column as an identity column.



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CREATE TABLE t1(     id INT GENERATED BY DEFAULT AS IDENTITY        (START WITH 10 INCREMENT BY 10),     c1 VARCHAR(10),     PRIMARY KEY(id)  ); |

The value of the id column will start with 10 and increment by 10.

Second, use the following [INSERT](https://www.db2tutorial.com/db2-basics/db2-insert/) statement to insert three rows into the t1 table:



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO     t1(c1)  VALUES     ('A'),('B'),('C'); |

Third, view data from the t1 table using the following SELECT statement:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM t1; |

Here is the output:



|  |  |
| --- | --- |
| 1  2  3  4  5 | ID          C1  ----------- ----------  10          A  20          B  30          C |

### 2) Db2 identity column with CYCLE example

First, create a new table named t2 whose id column is an identity column.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | CREATE TABLE t2(     id INT GENERATED ALWAYS AS IDENTITY        (START WITH -1,        INCREMENT BY 1,        CYCLE,        MINVALUE -1,        MAXVALUE 2),     c2 VARCHAR(10)  ); |

Second, insert seven rows into the t2 table:



|  |  |
| --- | --- |
| 1  2  3 | INSERT INTO     t2(c2)  VALUES('A'),('B'),('C'),('D'),('E'),('F'); |

Third, query data from the t2 table:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM t2; |

Here is the output:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | ID          C2  ----------- ----  -1          A  0           B  1           C  2           D  -1          E  0           F  1           G |

In this example, the id column’s value starts with -1 and has an increment of one.

Because the MAXVALUE is 2 and CYCLE option is specified, the sequence is -1, 0, 1 ,2, -1, 0, 1 …

# Db2 ALTER TABLE ALTER COLUMN

**Summary**: in this tutorial, you’ll learn how to use the Db2 ALTER TABLE ALTER COLUMN statement to modify column definitions.

## Introduction to Db2 ALTER TABLE ALTER COLUMN statement

Sometimes, you may want to modify column definition to cope with the new business requirements such as extend the maximum length of a column or changing the default value of a column to a new one.

Db2 ALTER TABLE ALTER COLUMN allows you to modify the definitions of the existing columns in a table.

The following illustrates the syntax of the ALTER TABLE ALTER COLUMN statement:



|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE table\_name      ALTER COLUMN column\_name      modification\_type; |

In this syntax:

* First, specify the name of the table which you want to perform the change in the ALTER TABLE clause.
* Second, specify the name of the column that you want to modify.
* Third, add a clause as a modification type. Db2 allows you to change the data type of a column, adjust the column length, and change the default value to a new one.

## Db2 ALTER TABLE ALTER COLUMN examples

We’ll use the orders table created in the [ALTER TABLE ADD COLUMN tutorial](https://www.db2tutorial.com/db2-basics/db2-alter-table-add-column/) for the demonstration.

### 1) Modifying the length & data type of column examples

The following example uses the ALTER TABLE ALTER COLUMN statement to change the data type of the created\_date column from DATE to TIMESTAMP:



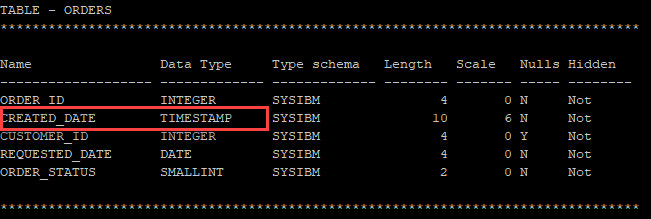
|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE orders      ALTER COLUMN created\_date      SET DATA TYPE TIMESTAMP; |

To view the columns of the orders table, you use the DESCRIBE TABLE command:



|  |  |
| --- | --- |
| 1 | DESCRIBE TABLE orders; |

Here is the output:



This statement [adds a new column](https://www.db2tutorial.com/db2-basics/db2-alter-table-add-column/) named note to the orders table:



|  |  |
| --- | --- |
| 1  2 | ALTER TABLE orders  ADD COLUMN note VARCHAR(40); |

To increase the length of the note column to 255, you use the following statement:



|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE orders      ALTER COLUMN note      SET DATA TYPE VARCHAR(255); |

### 2) Changing the default value to a new one example

To change the default value of a column to a new value, you use the following syntax:



|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE table\_name      ALTER COLUMN column\_name      SET DEFAULT new\_default\_value |

The following example sets the default value of the created\_date column to the current timestamp:



|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE orders      ALTER COLUMN created\_date      SET DEFAULT CURRENT\_TIMESTAMP; |

# Db2 ALTER TABLE ADD COLUMN

**Summary**: in this tutorial, you will learn how to add one or many columns to a table by using the Db2 ALTER TABLE ADD COLUMN statement.

## Introduction to Db2 ALTER TABLE ADD COLUMN statement

Because of the new business requirements, you may want to add one or more columns to an existing table.

To add a column to a table, you use the ALTER TABLE ADD COLUMN statement as shown in the following syntax:



|  |  |
| --- | --- |
| 1  2 | ALTER TABLE table\_name  ADD COLUMN column\_name data\_type column\_constraint; |

In this syntax:

* First, specify the name of the table to which you want to add the new column in the ALTER TABLE clause.
* Second, specify the new column including name, data type, and column constraint in the ADD COLUMN clause.

If you want to add many columns to an existing table, you use multiple ADD COLUMN clauses as shown the following syntax:



|  |  |
| --- | --- |
| 1  2  3  4  5 | ALTER TABLE     table\_name  ADD COLUMN c1 data\_type constraint  ADD COLUMN c2 data\_type constraint  ...; |

Note that there is no comma between the ADD COLUMN clauses.

## Db2 ALTER TABLE ADD COLUMN examples

Let’s take some examples of using the ALTER TABLE ADD COLUMN statement.

First, [create a new table](https://www.db2tutorial.com/db2-basics/db2-create-table/) named orders for demonstration.



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CREATE TABLE orders (     order\_id INT GENERATED ALWAYS        AS IDENTITY NOT NULL,     created\_date DATE NOT NULL,     PRIMARY KEY(order\_id)  ); |

### 1) Using Db2 ALTER TABLE ADD COLUMN to add one column example

To add a new column named customer\_id to the orders table, you use the following ALTER TABLE ADD COLUMN statement:



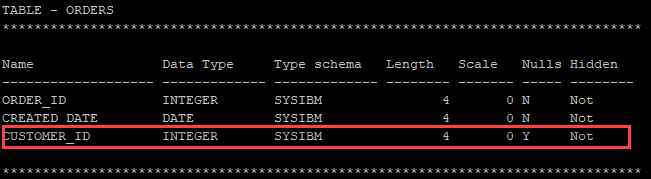
|  |  |
| --- | --- |
| 1  2 | ALTER TABLE orders  ADD COLUMN customer\_id INT; |

To verify the change to the columns of the orders table, you use the DESCRIBE TABLE command:



|  |  |
| --- | --- |
| 1 | DESCRIBE TABLE orders; |

Here is the output:



### 2) Using Db2 ALTER TABLE ADD COLUMN to add multiple columns example

The following example adds two new columns named requested\_date and order\_status to the orders table:

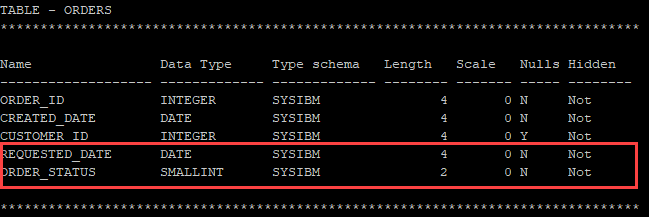


|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE orders  ADD COLUMN requested\_date DATE NOT NULL DEFAULT CURRENT\_DATE  ADD COLUMN order\_status SMALLINT NOT NULL DEFAULT 0; |

Here is the new structure of the orders table:



|  |  |
| --- | --- |
| 1 | DESCRIBE TABLE orders; |



# Db2 ALTER TABLE DROP COLUMN

**Summary**: in this tutorial, you’ll learn how to drop one or more columns in a table by using the Db2 ALTER TABLE DROP COLUMN statement.

## Introduction to Db2 ALTER TABLE DROP COLUMN statement

Sometimes, you may want to delete one or more unused columns from a table. To do this, you use the ALTER TABLE DROP COLUMN statement as follows:



|  |  |
| --- | --- |
| 1  2 | ALTER TABLE table\_name  DROP COLUMN column\_name; |

In this syntax:

* First, specify the name of the table from which you want to drop the column in the ALTER TABLE clause.
* Second, specify the name of the column that you want to delete in the DROP COLUMN clause.

To delete multiple columns, you can use the following syntax:

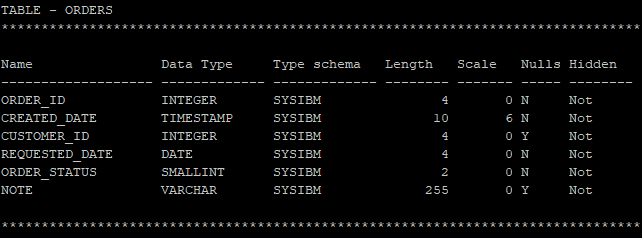


|  |  |
| --- | --- |
| 1  2  3  4 | ALTER TABLE table\_name  DROP COLUMN column\_name\_1  DROP COLUMN column\_name\_2  ...; |

Note that there is no comma between the DROP COLUMN clauses.

## Db2 ALTER TABLE DROP COLUMN examples

We’ll use the orders table created in the ALTER TABLE tutorial for the demonstration.



### 1) Dropping one column example

The following example removes the note column from the orders table:



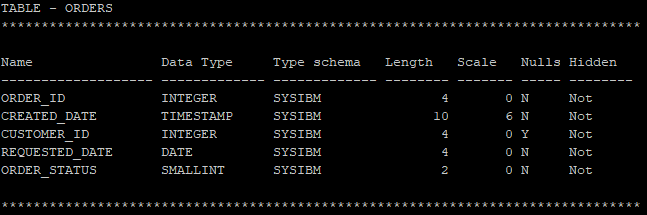
|  |  |
| --- | --- |
| 1  2 | ALTER TABLE orders  DROP COLUMN note; |

To verify the change, you can use the DESCRIBE TABLE command:



|  |  |
| --- | --- |
| 1 | DESCRIBE TABLE orders; |

Here is the output:



The note column has been dropped.

### 2) Dropping multiple columns example

This example uses the ALTER TABLE DROP COLUMN to drop the requested\_date and order\_status columns:

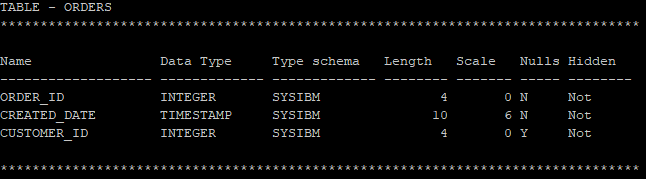


|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE orders  DROP COLUMN requested\_date  DROP COLUMN order\_status; |

The following picture shows the column list of the orders table after executing the above statement:



|  |  |
| --- | --- |
| 1 | DESCRIBE TABLE orders; |



# Db2 ALTER TABLE DROP COLUMN

**Summary**: in this tutorial, you’ll learn how to drop one or more columns in a table by using the Db2 ALTER TABLE DROP COLUMN statement.

## Introduction to Db2 ALTER TABLE DROP COLUMN statement

Sometimes, you may want to delete one or more unused columns from a table. To do this, you use the ALTER TABLE DROP COLUMN statement as follows:



|  |  |
| --- | --- |
| 1  2 | ALTER TABLE table\_name  DROP COLUMN column\_name; |

In this syntax:

* First, specify the name of the table from which you want to drop the column in the ALTER TABLE clause.
* Second, specify the name of the column that you want to delete in the DROP COLUMN clause.

To delete multiple columns, you can use the following syntax:

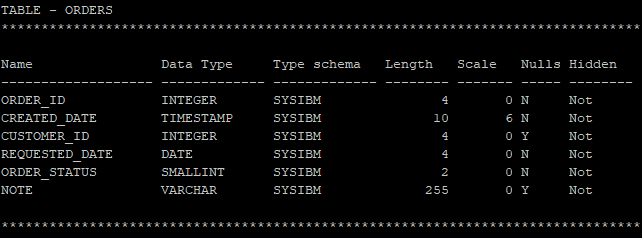


|  |  |
| --- | --- |
| 1  2  3  4 | ALTER TABLE table\_name  DROP COLUMN column\_name\_1  DROP COLUMN column\_name\_2  ...; |

Note that there is no comma between the DROP COLUMN clauses.

## Db2 ALTER TABLE DROP COLUMN examples

We’ll use the orders table created in the ALTER TABLE tutorial for the demonstration.



### 1) Dropping one column example

The following example removes the note column from the orders table:



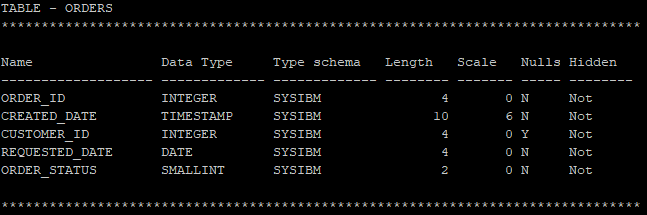
|  |  |
| --- | --- |
| 1  2 | ALTER TABLE orders  DROP COLUMN note; |

To verify the change, you can use the DESCRIBE TABLE command:



|  |  |
| --- | --- |
| 1 | DESCRIBE TABLE orders; |

Here is the output:



The note column has been dropped.

### 2) Dropping multiple columns example

This example uses the ALTER TABLE DROP COLUMN to drop the requested\_date and order\_status columns:

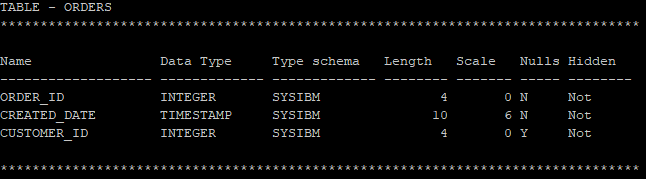


|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE orders  DROP COLUMN requested\_date  DROP COLUMN order\_status; |

The following picture shows the column list of the orders table after executing the above statement:



|  |  |
| --- | --- |
| 1 | DESCRIBE TABLE orders; |



# Db2 DROP TABLE

**Summary**: in this tutorial, you’ll learn how to use the Db2 DROP TABLE statement to delete an existing table from a database.

## Introduction to Db2 DROP TABLE statement

Sometimes, you may want to delete one or more unused tables from a database. To do this, you use the DROP TABLE statement as follows:



|  |  |
| --- | --- |
| 1 | DROP TABLE [schema\_name.]table\_name; |

In this syntax:

* First, specify the name of the schema to which the table belongs. The schema is optional. If you skip it, the statement will delete the specified table in the current schema.
* Second, specify the name of the table that you want to drop.

When you use the DROP TABLE statement to delete a table, Db2 performs the following actions:

* Delete all data in the table permanently.
* Delete all columns of the dropped table and the indexes associated with these columns.
* Mark all views that reference to the dropped table as inoperative.
* Also, mark all triggers that depend on the dropped table as inoperative.
* Revoke all privileges on the table and dependent views.

## Db2 DROP TABLE examples

Let’s [create three new tables](https://www.db2tutorial.com/db2-basics/db2-create-table/) to demonstrate the DROP TABLE statement.



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | CREATE TABLE t1(     id INT NOT NULL PRIMARY KEY  );    CREATE TABLE t2(     id INT NOT NULL PRIMARY KEY  );    CREATE TABLE t3(     id INT NOT NULL PRIMARY KEY,     fk INT NOT NULL,     FOREIGN KEY fk\_t2(fk) REFERENCES t2(id)  ); |

### 1) Drop a table example

The following statement uses the DROP TABLE statement to drop the t1 table:



|  |  |
| --- | --- |
| 1 | DROP TABLE t1; |

This example is straight forward. The table t1 was just dropped successfully.

In practice, a table may have [foreign key](https://www.db2tutorial.com/db2-basics/db2-foreign-key/) references like the t2 table.

### 2) Drop a table with foreign key constraint example

The following example uses the DROP TABLE statement to drop the t2 table:



|  |  |
| --- | --- |
| 1 | DROP TABLE t2; |

The t2 table was dropped. In addition, the foreign key fk\_t2 was marked as inoperative.

# Db2 TRUNCATE TABLE

**Summary**: in this tutorial, you will learn how to use the Db2 TRUNCATE TABLE statement to delete all rows from a table.

## Introduction to Db2 TRUNCATE TABLE statement

The [DELETE](https://www.db2tutorial.com/db2-basics/db2-delete/) statement without a [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause allows you to delete all rows from a table. However, if you have a table with a large volume of data, the DELETE statement will not be efficient.

Fortunately, Db2 provides the TRUNCATE TABLE that also delete all rows from a table, but in a more efficient way.

The following shows the syntax of the TRUNCATE TABLE statement:



|  |  |
| --- | --- |
| 1  2  3  4 | TRUNCATE TABLE table\_name  [DROP STORAGE | REUSE STORAGE]  [IGNORE DELETE TRIGGERS | RESTRICT WHEN DELETE TRIGGERS]  [IMMEDIATE] |

In this syntax:

First, specify the name of the table from which you want to delete all data.

Second, the DROP STORAGE or REUSE STORAGE option determines whether Db2 should drop or reuse the existing storage allocated for the table.

* The DROP STORAGE releases all storages allocated for the table and make the storage available for use for the same table or any other tables within the tablespace.
* The REUSE STORAGE makes all storages empty, but continues allocating for the table.

The default option is DROP STORAGE.

Third, the IGNORE DELETE TRIGGERS and RESTRICT WHEN DELETE TRIGGERS options determine the behavior of the truncate operation when the table has associated delete triggers.

* The IGNORE DELETE TRIGGERS option will not fire any trigger defined on the table.
* The RESTRICT WHEN DELETE TRIGGERS option will result in an error if the table has one or more deleted triggers.

By default, the TRUNCATE TABLE statement uses IGNORE DELETE TRIGGERS option.

Fourth, the IMMEDIATE option, if specified, will delete all rows from the table without the ability to roll back. On the other hand, if you don’t specify the IMMEDIATE option, you can use a ROLLBACK statement to undo the truncate operation.

Notice that when you use the TRUNCATE TABLE statement in a transaction, you must place it as the first statement.

## Db2 TRUNCATE TABLE examples

Let’s take some examples of using the TRUNCATE TABLE statement.

First, [create a new table](https://www.db2tutorial.com/db2-basics/db2-create-table/) named books2 that has the same structure as the books table:



|  |  |
| --- | --- |
| 1 | CREATE TABLE books2 LIKE books; |

Then, insert all rows from the books table to the books2 table using the [INSERT INTO SELECT](https://www.db2tutorial.com/db2-basics/db2-insert-into-select/) statement:



|  |  |
| --- | --- |
| 1  2 | INSERT INTO books2  SELECT \* FROM books; |

### 1) Using Db2 TRUNCATE TABLE statement within a transaction example

First, start a new transaction:



|  |  |
| --- | --- |
| 1 | BEGIN TRANSACTION; |

Second, truncate the book2 table:



|  |  |
| --- | --- |
| 1 | TRUNCATE TABLE books2; |

Third, rollback the transaction:



|  |  |
| --- | --- |
| 1 | ROLLBACK; |

Fourth, verify the truncate operation by querying some rows from the books2 table:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT     title  FROM     books2  ORDER BY     title  LIMIT 5; |

Here is the output:

As you can see clearly from the output, the truncate operation on the books2 table was rolled back successfully.

### 2) Using Db2 TRUNCATE TABLE statement to truncate a table immediately

The following example uses the TRUNCATE TABLE statement to truncate the books2 table immediately with the IMMEDIATE option:



|  |  |
| --- | --- |
| 1  2 | TRUNCATE TABLE books2  IMMEDIATE; |

If you query data from the books2 table, you will get an empty result set:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM books2; |

# Db2 RENAME TABLE

**Summary**: in this tutorial, you will learn how to use the Db2 RENAME TABLE statement to change the name of a table to the new one.

## Introduction to Db2 RENAME TABLE statement

To change the name of an existing table to a new one, you use the RENAME TABLE statement.

Here is the syntax of the RENAME TABLE statement:



|  |  |
| --- | --- |
| 1 | RENAME TABLE table\_name TO new\_table\_name; |

In this syntax:

* First, specify the name of the table to which you want to rename after the RENAME TABLE keywords.
* Then, specify the new name of the table after the TO keyword.

Db2 requires that the table, which you want to rename, must meet the following requirements:

* The table must not have any reference in existing objects such as view, triggers, functions, or constraints.
* The table must not have any generated column, other than the [identity column](https://www.db2tutorial.com/db2-basics/db2-identity-column/), or be a parent or dependent table.

Notice that this behavior is different from other database systems such as [Oracle](https://www.oracletutorial.com/oracle-basics/oracle-rename-table/) and [SQL Server](http://www.sqlservertutorial.net/sql-server-basics/sql-server-rename-table/), which mark the dependent objects of the table invalid.

## Db2 RENAME TABLE statement example

First, [create a new table](https://www.db2tutorial.com/db2-basics/db2-create-table/) named favorites:



|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE favorites (     book\_id INT NOT NULL,     user\_id INT NOT NULL,     PRIMARY KEY (book\_id,user\_id)  ); |

Then, use the RENAME TABLE statement to change the favorites table to favorite\_books table:



|  |  |
| --- | --- |
| 1 | RENAME TABLE favorites TO favorite\_books; |

# Db2 Primary Key

**Summary**: in this tutorial, you will learn how to use the Db2 PRIMARY KEY constraint to define a primary key for a table.

## Db2 primary key overview

A primary key of a table is a column or group of columns whose values uniquely identify every row in the table.

Each table has one and only one primary key. A primary key is optional. However, it’s a good practice to have a primary key in every table.

To define a primary key for a table, you use the PRIMARY KEY constraint.

### Creating a table with a primary key

The following statement [creates a table](https://www.db2tutorial.com/db2-basics/db2-create-table/) with a primary key:



|  |  |
| --- | --- |
| 1  2  3  4 | CREATE TABLE table\_name (     pk\_column type NOT NULL PRIMARY KEY,     ...  ); |

If the primary key consists of two or more columns, you use the following syntax:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE TABLE table\_name (     pk\_column1 type NOT NULL,     pk\_column2 type NOT NULL,     ...,     PRIMARY KEY(pk\_column1,pk\_column2,...)  ); |

When you create a table with the primary key constraint, Db2 automatically creates a unique index on the primary key columns. This unique index is also known as the primary index.

### Adding a primary key to an existing table

To add a primary key to a table, you use the following form of ALTER TABLE statement:



|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE table\_name     [ADD CONSTRAINT constraint\_name]       PRIMARY KEY (primary\_key\_columns); |

Note that the ADD CONSTRAINT clause is optional.

### Removing a primary key from a table

To remove a primary key from a table, you use the following form of the ALTER TABLE statement:



|  |  |
| --- | --- |
| 1  2 | ALTER TABLE table\_name     DROP PRIMARY KEY; |

### Primary key and NOT NULL constraint

The primary key of a table cannot contain NULL values. In other database systems such as SQL Server, Oracle, and MySQL, the columns included in the primary key columns will implicitly receive the NOT NULL constraint. However, in DB2, you must explicitly declare all primary key columns as NOT NULL columns.

### Primary key and identity column

Because the primary key must contain unique values that uniquely identify each row of a table, the [identity column](https://www.db2tutorial.com/db2-basics/db2-identity-column/) is an ideal choice for the primary key.

The following syntax shows how to create a table with a primary key as an identity column:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE TABLE table\_name(     pk\_column INT        GENERATED ALWAYS AS IDENTITY        NOT NULL        PRIMARY KEY,     ...,  ); |

## Db2 primary key examples

Let’s take some examples of creating new tables with primary keys.

### 1) Creating a table with a primary key example

Suppose [SSN](https://en.wikipedia.org/wiki/Social_Security_number) uniquely identifies a person. The following statement creates the persons table whose primary key is the SSN column:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CREATE TABLE persons (     ssn VARCHAR(11) NOT NULL PRIMARY KEY,     first\_name VARCHAR(100) NOT NULL,     last\_name VARCHAR(100) NOT NULL,     gender CHAR(1)  ); |

This statement [inserts](https://www.db2tutorial.com/db2-basics/db2-insert/) a new person into the persons table:



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO     persons(ssn,first\_name,last\_name,gender)  VALUES     ('123-45-6789','John','Doe','M'); |

The following statement attempts to [insert a new row](https://www.db2tutorial.com/db2-basics/db2-insert/) into the persons table with an SSN that already exists:



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO     persons(ssn,first\_name,last\_name,gender)  VALUES     ('123-45-6789','Jane','Doe','F'); |

Db2 issued the following error:



|  |  |
| --- | --- |
| 1  2  3  4 | SQL0803N  One or more values in the INSERT statement, UPDATE statement, or  foreign key update caused by a DELETE statement are not valid because the  primary key, unique constraint or unique index identified by "1" constrains  table "DB2ADMIN.PERSONS" from having duplicate values for the index key. |

### 2) Creating a table with a primary key as an identity column example

The following example creates a table named members with a primary key column as an identity column:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | CREATE TABLE members(     member\_id INT        GENERATED BY DEFAULT AS IDENTITY        PRIMARY KEY,     first\_name VARCHAR(100) NOT NULL,     last\_name VARCHAR(100) NOT NULL,     gender CHAR(1)  ); |

Each time you insert a new row into the members table, Db2 automatically generates a sequential number for the member\_id column.



|  |  |
| --- | --- |
| 1  2  3  4  5 | INSERT INTO     members(first\_name,last\_name,gender)  VALUES     ('Mary','Jane','F'),     ('Peter','Crane','M'); |

Here are the contents of the members table:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM members; |

Db2 Primary Key as Identity column

Notice that the member\_id has the sequential integer 1 and 2.

### 3) Creating a new table whose primary key has more than one column example

The following example creates a new table whose primary key contains two columns:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CREATE TABLE enrollments (     member\_id INT NOT NULL,     membership\_id INT NOT NULL,     start\_date DATE NOT NULL,     PRIMARY KEY(member\_id, membership\_id)  ); |

### 4) Adding a primary key to an existing table example

This example creates a new table without a primary key:



|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE memberships(     membership\_id INT NOT NULL,     name VARCHAR(100) NOT NULL,     description VARCHAR(150)  ); |

Before promoting the membership\_id as the primary key, you first need to make sure that this column does not have duplicate values. Then, you use the following statement to add a primary key to the memberships table:



|  |  |
| --- | --- |
| 1  2 | ALTER TABLE memberships     ADD PRIMARY KEY(membership\_id); |

# Db2 Foreign Key

**Summary**: in this tutorial, you will learn how to use the Db2 foreign key constraint to enforce the referential integrity between the data across tables.

## Introduction to the Db2 foreign key

Let’s take a look at the contacts and phones tables:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | CREATE TABLE contacts(     contact\_id INT NOT NULL GENERATED ALWAYS AS IDENTITY,     first\_name VARCHAR(100) NOT NULL,     last\_name VARCHAR(100) NOT NULL,     PRIMARY KEY(contact\_id)  );    CREATE TABLE phones(     phone\_id INT NOT NULL GENERATED ALWAYS AS IDENTITY,     phone\_no VARCHAR(20) NOT NULL,     phone\_type VARCHAR(10) NOT NULL,     contact\_id INT NOT NULL,     PRIMARY KEY(phone\_id)  ); |

In this diagram, each contact may have zero or many phones such as home phone, work phone, and emergency phone. However, each phone must belong to one and only one contact. The relationship between the contacts and phones is one-to-many.

For each row in the phones table, you can always find a corresponding row in the contacts table. But the current setup does not ensure this relationship. It means you can insert a new row into the phones table with the contact identification (contact\_id) that does not exist in the contacts table.

Furthermore, if you delete a contact, all the phones of the deleted contact will remain in the phones table. The rows in the phones table that does not have corresponding rows in the contacts table are called orphaned rows.

In order to enforce the relationship between contacts and phones tables, you need to use a foreign key constraint.

### What is a foreign key?

A foreign key is a column or group of columns in a table that uniquely identifies a row in another table. The foreign key constraints define foreign keys.

Back to our example, the contact\_id in the phones table should be the foreign key of the phones table. Because for each phone in the phones table, you can find a corresponding contact in the contacts table.

To add a foreign key constraint to the phones table, you use the following ALTER TABLE statement:



|  |  |
| --- | --- |
| 1  2  3  4  5 | ALTER TABLE phones  FOREIGN KEY (contact\_id)     REFERENCES contacts(contact\_id)        ON UPDATE NO ACTION        ON DELETE CASCADE; |

The contacts table is called the parent table to which the foreign key references. The phones table is called the child table (or dependent table) to which the foreign key constraint is applied.

The contact\_id column in the contacts table is called the parent key and the contact\_id column in the phones table is called the foreign key or foreign key column.

In the database world, referential integrity is a mechanism to ensure that the relationship of data between tables remains consistent. And to enforce the referential integrity, you use foreign key constraints. Therefore, foreign key constraints are also known as referential integrity constraints or referential constraints.

## Db2 FOREIGN KEY constraint syntax

The following illustrates the syntax of defining a foreign key constraint:



|  |  |
| --- | --- |
| 1  2  3  4  5 | [CONSTRAINT constraint\_name]  FOREIGN KEY (fk1, fk2,...)  REFERENCES parent\_table(c1,2,..)     ON UPDATE [ NO ACTION | RESTRICT]     ON DELETE [ NO ACTION | RESTRICT | CASCADE | SET NULL]; |

In this syntax:

First, specify a constraint name in the CONSTRAINT clause. The CONSTRAINT clause is optional. If you omit it, Db2 will generate a name for the foreign key constraint.

Second, specify a list of comma-separated foreign key columns enclosed by parentheses in the FOREIGN KEY clause.

Third, specify the name of the parent table and a list of comma-separated columns to which the foreign key columns reference.

### ON UPDATE rules

Db2 triggers the ON UPDATE rule when you update a row in either parent or child table. The update rule has two options NO ACTION and RESTRICT.

When you update the row in the parent key column of the parent table, Db2 rejects the update if there is the corresponding row exists in the child table for both RESTRICT and NO ACTION option.

When you update the row in the foreign key column of the child table, Db2 rejects the update for RESTRICT option and allows the update for the NO ACTION, with the condition that the new value of the foreign key column exists in the parent table.

### ON DELETE rules

Db2 triggers the ON DELETE rule when you delete a row in the parent table. Db2 determines whether or not to delete the rows in the child table based on the following options:

* NO ACTION or RESTRICT does not delete any row in both tables and issues an error.
* CASCADE deletes the row in the parent table and all related rows in the child table.
* SET NULL deletes the row in the parent table and updates values in the foreign key columns in the child table to NULL only if these columns are not nullable columns.

You can use the foreign key constraint to define foreign keys in the CREATE TABLE or ALTER TABLE statement.

## Db2 FOREIGN KEY constraint examples

Let’s take some example of using the foreign key constraint to understand it better.

### 1) Creating a table which has a single foreign key example

First, [insert](https://www.db2tutorial.com/db2-basics/db2-insert/) a new contact into the contacts table:



|  |  |
| --- | --- |
| 1  2 | INSERT INTO contacts(first\_name, last\_name)  VALUES('John','Doe'); |

The contact John Doe has the contact id 1:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM contacts; |

Next, add two phones for the contact John Doe:



|  |  |
| --- | --- |
| 1  2  3  4  5 | INSERT INTO     phones(phone\_no, phone\_type, contact\_id)  VALUES     ('(408)-987-1234','HOME',1),     ('(408)-672-3424','WORK',1); |

Then, [delete](https://www.db2tutorial.com/db2-basics/db2-delete/) the contact id 1 from the contacts table. Because we declare the ON DELETE rule with the CASCADE action, Db2 will delete all phones of John Doe from the phones table:



|  |  |
| --- | --- |
| 1  2 | DELETE FROM contacts  WHERE contact\_id = 1; |

After, verify the deletion by querying data from the contacts table:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM contacts; |

It returned no row.

Finally, view data in the contacts table by using the following [SELECT](https://www.db2tutorial.com/db2-basics/db2-select/) statement:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM phones; |

It also returns an empty set.

### 2) Creating a table with multiple foreign keys example

The following statement [creates a new table](https://www.db2tutorial.com/db2-basics/db2-create-table/) called members:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE TABLE members (     member\_id INT NOT NULL GENERATED ALWAYS AS IDENTITY,     first\_name VARCHAR(100) NOT NULL,     last\_name VARCHAR(100) NOT NULL,     joined\_date DATE NOT NULL,     PRIMARY KEY(member\_id)  ); |

Suppose each member can have one or many favorite books and each book may belong to favorite lists of many users. The relationship between members and books are many-to-many.

The following favorite\_books table stores the favorite books of members:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | CREATE TABLE favorite\_books(     member\_id INT NOT NULL,     book\_id INT NOT NULL,     FOREIGN KEY (book\_id)        REFERENCES books(book\_id)        ON UPDATE RESTRICT        ON DELETE CASCADE,     FOREIGN KEY (member\_id)        REFERENCES members(member\_id)        ON UPDATE RESTRICT        ON DELETE CASCADE  ); |

The favorite\_books table has two foreign keys. The first one refers to the book\_id column of the books table and the second one references to the member\_id of the members table.

The favorite\_books table is known as an associative table, pivot table, or mapping table. We often use these kinds of tables to manage the many-to-many relationship.

# Db2 DEFAULT Constraint

**Summary**: in this tutorial, you will learn how to use the Db2 DEFAULT constraint to specify a value inserted into a column if the application does not provide the value.

## Introduction to Db2 DEFAULT constraint

The DEFAULT constraint specifies a predefined value inserted into a column if the application doesn’t supply a value.

The following illustrates how to add a DEFAULT constraint to a column when you [create a new table](https://www.db2tutorial.com/db2-basics/db2-create-table/):



|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE table\_name (      ...,      column\_name type DEFAULT default\_value      ...  ); |

In this syntax, you need to specify the DEFAULT keyword following by the default value. The default value depends on the data type of the column.

This table illustrates the data types and their corresponding default values:

| **Column data type** | **Default value** |
| --- | --- |
| SMALLINT, [INTEGER](https://www.db2tutorial.com/db2-basics/db2-integer/), DECIMAL, NUMERIC, NUM, REAL, FLOAT, DOUBLE, DOUBLE RECISION | 0 |
| [CHAR](https://www.db2tutorial.com/db2-basics/db2-char/), GRAPHIC | A string of blank characters |
| [VARCHAR](https://www.db2tutorial.com/db2-basics/db2-varchar/), LONG VARCHAR, VARGRAPHIC, LONG VARGRAPHIC, BLOB, CLOB, DBCLOB | A zero-length string |
| [DATE](https://www.db2tutorial.com/db2-basics/db2-date/) | The current time( CURRENT\_DATE) when the row is inserted into the table. Existing rows receive the date 0001-01-01 |
| [TIME](https://www.db2tutorial.com/db2-basics/db2-time/) | The current time( CURRENT\_TIME) when the row is inserted into the table. Existing rows receive the time 00:00:00 |
| [TIMESTAMP](https://www.db2tutorial.com/db2-basics/db2-timestamp/) | The current timestamp ( CURRENT\_TIMESTAMP) when the row is inserted into the table. |
| XML | Not applicable |
| user-defined data type | The default value of the built-in data type on which the distinct user-defined data type is based |

Db2 applies the default constraints when new rows are inserted into the table via INSERT, IMPORT, LOAD or INGEST commands.

If the column does not have the NOT NULL constraint. Its default value is NULL.

## Db2 DEFAULT constraint example

First, [create a new table](https://www.db2tutorial.com/db2-basics/db2-create-table/) named reviews to store book reviews:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | CREATE TABLE reviews(      review\_id INT NOT NULL          GENERATED ALWAYS AS IDENTITY,      book\_id INT NOT NULL,      user\_id INT NOT NULL,      review VARCHAR(255) NOT NULL,      review\_at TIMESTAMP NOT NULL          DEFAULT CURRENT\_TIMESTAMP,      FOREIGN KEY (book\_id)          REFERENCES books(book\_id),      PRIMARY KEY(review\_id)  ); |

In the reviews table, the review\_at has a default value as the CURRENT\_TIMESTAMP. It means that if you don’t supply a value to the review\_at column, the Db2 will use the current time to insert into this column.

Second, [insert](https://www.db2tutorial.com/db2-basics/db2-insert/) a new review to the reviews table:



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO      reviews(book\_id, user\_id, review)  VALUES      (1,1,'This book is excellent that contains many useful information'); |

Third, query data from the reviews table:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | SELECT      review\_id,      book\_id,      user\_id,      review,      review\_at  FROM      reviews; |

Here is the output:

Db2 Default Constraint example

As you can see clearly from the output, the review\_at column received the current timestamp.

## Adding the DEFAULT constraint to a column

To add the default constraint to an existing column, you use the following ALTER TABLE statement:



|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE table\_name  ALTER COLUMN column\_name  SET DEFAULT default\_value; |

Consider the following example:

First, add the rating column to the reviews table:



|  |  |
| --- | --- |
| 1  2 | ALTER TABLE reviews  ADD COLUMN rating SMALLINT; |

Then, add a default constraint to the rating column:



|  |  |
| --- | --- |
| 1  2 | ALTER TABLE reviews  ALTER COLUMN rating SET DEFAULT 3; |

If you want to add a new column with a default value, you can do it in one step:



|  |  |
| --- | --- |
| 1  2 | ALTER TABLE reviews  ADD COLUMN RATING SMALLINT DEFAULT 3; |

# Db2 NOT NULL

**Summary**: in this tutorial, you will learn how to use the Db2 NOT NULL constraint to prevent NULL values stored in a column.

## Db2 NOT NULL constraint overview

In the database world, NULL is a marker or special value that indicates the missing information or the information is not applicable.

To avoid NULL to be stored in a column, you use the NOT NULL constraint with the following syntax:



|  |  |
| --- | --- |
| 1 | column\_name type NOT NULL |

Unlike [primary key](https://www.db2tutorial.com/db2-basics/db2-primary-key/) and [foreign key](https://www.db2tutorial.com/db2-basics/db2-foreign-key/) constraints, the NOT NULL constraints are the column constraints, therefore, they must appear in the column definitions of a table.

If you declare a column without specifying the NOT NULL constraint, Db2 will assume that the column accepts NULL values. To declare a column that explicitly accepts NULL values, you use the following syntax:



|  |  |
| --- | --- |
| 1 | column\_name type NULL |

### Db2 NOT NULL constraint example

The following statement [creates a new table](https://www.db2tutorial.com/db2-basics/db2-create-table/) named discounts that stored discounted books:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12 | CREATE TABLE discounts (     book\_id INT NOT NULL,     start\_date DATE NOT NULL,     end\_date DATE NOT NULL,     percentage DECIMAL(4,2) NOT NULL,     description VARCHAR(50),     PRIMARY KEY(book\_id, start\_date, end\_date),     FOREIGN KEY(book\_id)        REFERENCES books(book\_id)           ON UPDATE RESTRICT           ON DELETE CASCADE  ); |

In the discounts table, all the columns have NOT NULL constraints except for the description column.

Even though the book\_id, start\_date, end\_date are the primary key columns, you need to include the NOT NULL constraint because the primary key does not accept NULL.

Typically, in other database systems, when you define the primary key of a table, all primary key columns implicitly have the NOT NULL constraints.

## Adding NOT NULL constraint to an existing column

Sometimes, you may need to make a nullable column NOT NULL. To do this follow these steps:

First, update the value of the column to non-NULL values:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | UPDATE     table\_name  SET     column\_name = non-null value  WHERE     column\_name IS NULL; |

Then, modify the column to add theNOT NULL constraint:



|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE table\_name  ALTER COLUMN column\_name  SET NOT NULL; |

For example, to make the description column of the discounts table NOT NULL, you use these steps:

First, update the description to blank if the value in this column is NULL:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | UPDATE     discounts  SET     description = ''  WHERE     description IS NULL; |

Then, modify the description column using the following ALTER TABLE statement:



|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE discounts  ALTER COLUMN description  SET NOT NULL; |

## Removing the NOT NULL constraint

You can use the following [ALTER TABLE ALTER COLUMN](https://www.db2tutorial.com/db2-basics/db2-alter-table-alter-column/) statement to remove the NOT NULL constraint from a column:



|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE table\_name  ALTER COLUMN column\_name  DROP NOT NULL; |

The following statement removes the NOT NULL constraint from the description column of the discounts table:



|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE discounts  ALTER COLUMN description  DROP NOT NULL; |

# Db2 CHECK Constraint

**Summary**: in this tutorial, you will learn how to use the Db2 CHECK constraint to enforce data integrity in a table.

## Db2 CHECK constraint overview

A check constraint is a rule that specifies what values are allowed to be stored in a table. For example, you can specify that the price of a book must be positive.

To define a check constraint for a column, you use the following syntax:



|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE table\_name(     ...     column\_name type CHECK(Boolean\_expression),     ...  ); |

When you insert or update value into the column\_name, Db2 evaluates the Boolean\_expression of the check constraint. If the Boolean\_expression returns false, Db2 will reject the operation; otherwise, Db2 will allow.

Sometimes, you may want to check values across columns of a table. To do this, you use the following syntax:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CREATE TABLE table\_name(     column\_name1 type,     column\_name2 type,     ...,     CHECK(Boolean\_expression)  ); |

The Boolean\_expression in this syntax may include multiple columns of the table.

Note that the Boolean expression specified in the CHECK constraint only can include the columns within the same table. It cannot have columns from other tables.

To add a check constraint to a table, you use the following syntax:



|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE table\_name  ADD CONSTRAINT constraint\_name  CHECK(Boolean\_expression); |

To remove a check constraint from a table, you use this syntax:



|  |  |
| --- | --- |
| 1  2 | ALTER TABLE table\_name  DROP CONSTRAINT constraint\_name; |

## Db2 CHECK constraint examples

Let’s take some examples of using the Db2 CHECK constraint.

### 1) Creating a CHECK constraint when creating the table example

First, [create](https://www.db2tutorial.com/db2-basics/db2-create-table/) the book\_prices table that stores the book prices:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | CREATE TABLE book\_prices(     book\_id INT NOT NULL,     valid\_from DATE NOT NULL,     price DEC(10,2) CHECK(price > 0),     PRIMARY KEY(book\_ID, valid\_from),     FOREIGN KEY(book\_id)        REFERENCES books(book\_id)           ON UPDATE RESTRICT           ON DELETE CASCADE  ); |

In this example, the price column has a check constraint that allows only positive price to be inserted or updated to the column.

Second, [insert a new row](https://www.db2tutorial.com/db2-basics/db2-insert/) into the books\_price table whose price is positive.



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO     book\_prices(book\_id, valid\_from, price)  VALUES     (1, '2019-05-06',29.99); |

The statement works because the price is positive.

Third, attempt to insert a zero price into the book\_prices table:



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO     book\_prices(book\_id, valid\_from, price)  VALUES     (2, '2019-05-06',0); |

Db2 issued the following error because the price does not pass the check:



|  |  |
| --- | --- |
| 1  2 | SQL0545N  The requested operation is not allowed because a row does not satisfy  the check constraint "DB2ADMIN.BOOK\_PRICES.SQL190605115402090". |

Note that SQL190605115402090 is the constraint name generated automatically by Db2.

### 2) Adding a CHECK constraint to an existing table example

First, [add a new column](https://www.db2tutorial.com/db2-basics/db2-alter-table-add-column/) named to the book\_prices table using the [ALTER TABLE ADD COLUMN](https://www.db2tutorial.com/db2-basics/db2-alter-table-add-column/) statement:



|  |  |
| --- | --- |
| 1  2 | ALTER TABLE book\_prices  ADD COLUMN cost DEC(10,2); |

Second, add a check constraint to make sure that the cost is greater than zero and the price is always greater than or equal to the cost:



|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE book\_prices  ADD CONSTRAINT cost\_check  CHECK(cost > 0 AND price >= cost); |

Third, insert a row into the book\_prices table:



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO     book\_prices(book\_id, valid\_from, cost, price)  VALUES     (3, '2019-05-15',15.99, 20.99); |

The statement works because both cost and price pass the condition in the check constraint.

Fourth, insert a row whose cost is greater than the price:



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO     book\_prices(book\_id, valid\_from, cost, price)  VALUES     (4, '2019-05-15',16.99, 12.99); |

Db2 issued the following error:



|  |  |
| --- | --- |
| 1  2 | SQL0545N  The requested operation is not allowed because a row does not satisfy  the check constraint "DB2ADMIN.BOOK\_PRICES.COST\_CHECK". |

Fifth, insert a row whose cost is NULL:



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO     book\_prices(book\_id, valid\_from, cost, price)  VALUES     (5, '2019-05-15',NULL, 13.99); |

It worked because the condition cost > 0 AND price >= cost evaluates to unknown when the cost is NULL.

To avoid this kind of bypassing the check constraint, you need to use the NOT NULL constraint for the cost column.

### 3) Removing a CHECK constraint example

The following statement removes the cost\_check check constraint from the the book\_prices table:



|  |  |
| --- | --- |
| 1  2 | ALTER TABLE book\_prices  DROP CONSTRAINT cost\_check; |

# Db2 UNIQUE Constraint

**Summary**: in this tutorial, you will learn how to use the Db2 UNIQUE constraint to ensure the uniqueness of values stored in a column or a group of columns.

## Introduction to Db2 UNIQUE constraint

Db2 UNIQUE constraints ensure that the values in a column are unique and [not null](https://www.db2tutorial.com/db2-basics/db2-not-null/) for all rows in the table. For example, you may want the email is unique for every member.

To define a unique constraint for a column of a table, you use the following one of the following syntaxes:



|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE table\_name(     ...,     column\_name type UNIQUE,     ...  ); |

or



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CREATE TABLE table\_name(     ...,     column\_name type,     ...,     CONSTRAINT name UNIQUE(column\_name)  ); |

Behind the scenes, Db2 creates a unique index to enforce the uniqueness of the value in the column\_name.

Let’s take an example of using a unique constraint.

### Db2 UNIQUE constraint example

First, [create](https://www.db2tutorial.com/db2-basics/db2-create-table/) the members table whose email column will contain unique email addresses:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE TABLE members (     member\_id INT GENERATED BY DEFAULT AS IDENTITY NOT NULL ,     first\_name VARCHAR(100) NOT NULL,     last\_name VARCHAR(100) NOT NULL,     email VARCHAR(200) NOT NULL UNIQUE,     PRIMARY KEY(member\_id)  ); |

Second, [insert a new row](https://www.db2tutorial.com/db2-basics/db2-insert/) into the members table:



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO     members(first\_name, last\_name, email)  VALUES     ('John','Doe','john.doe@example.com'); |

Third, attempt to insert a new row with an email that already exists in the members table:



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO     members(first\_name, last\_name, email)  VALUES     ('John Joker','Doe','john.doe@example.com'); |

Db2 issued the following error due to unique constraint violation:



|  |  |
| --- | --- |
| 1  2  3  4 | SQL0803N  One or more values in the INSERT statement, UPDATE statement, or  foreign key update caused by a DELETE statement are not valid because the  primary key, unique constraint or unique index identified by "2" constrains  table "DB2ADMIN.MEMBERS" from having duplicate values for the index key. |

## Creating UNIQUE constraints for a group of columns

To define a unique constraint for a group of columns, you use the following syntax:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | CREATE TABLE table\_name(     ...,     column\_name\_1 type NOT NULL,     column\_name\_2 type NOT NULL,     ...,     UNIQUE(column\_name\_1,column\_name\_2,...)  ); |

In this case, Db2 will create a unique index that includes all the columns in the unique constraint to enforce the uniqueness. It will use the combination of values in the columns to evaluate duplicates.

## Adding UNIQUE constraints to an existing table

Before adding a unique constraint to a column or a group of columns, first, you must ensure that the current data is unique for all rows in the table.

Then, you can use the following statement to add a unique constraint to a table:



|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE table\_name  ADD CONSTRAINT constraint\_name  UNIQUE(column\_name\_1, column\_name\_2,...); |

Suppose, you want the combination of first name and last name of members to be unique. To achieve this, you use the following statement:



|  |  |
| --- | --- |
| 1  2  3 | ALTER TABLE members  ADD CONSTRAINT name  UNIQUE(first\_name,last\_name); |

The following statement inserts a new row into the members table:



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO     members(first\_name, last\_name, email)  VALUES     ('Jane','Doe','jane.doe@example.com'); |

The statement works because the combination of the first name and last name is not duplicate:



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO     members(first\_name, last\_name, email)  VALUES     ('Jane','Doe','jane.doe2@example.com'); |

Here is the error message:



|  |  |
| --- | --- |
| 1  2  3  4 | SQL0803N  One or more values in the INSERT statement, UPDATE statement, or  foreign key update caused by a DELETE statement are not valid because the  primary key, unique constraint or unique index identified by "3" constrains  table "DB2ADMIN.MEMBERS" from having duplicate values for the index key. |

## Removing UNIQUE constraints

The ALTER TABLE DROP CONSTRAINT statement removes the UNIQUE constraint:



|  |  |
| --- | --- |
| 1  2 | ALTER TABLE table\_name  DROP CONSTRAINT constraint\_name; |

For example, this statement removes the name constraint from the members table:



|  |  |
| --- | --- |
| 1  2 | ALTER TABLE members  DROP CONSTRAINT name; |

# Db2 Integers

**Summary**: in this tutorial, you will learn about various Db2 integer data types including SMALLINT, INT, and BIGINT.

## Introduction to Db2 integer types

Db2 supports three types of integers: SMALLINT, INT, and BIGINT.

### SMALLINT

SMALLINT is used to stores small integers with a precision of 15 bits. The range of SMALLINT is -32,768 to +32,767.

### INT

INT is used to store large integers with a precision of 31 bits. The range of INT is -2,147,483,648 to +2,147,483,647. INTEGER is the synonym of INT, therefore, they are interchangeable.

### BIGINT

BIGINT is used to store big integers with a precision of 63 bits. The range of big integers is -9,223,372,036,854,775,808 to +9,223,372,036,854,775,807.

When you design database tables, it is a good practice to choose the appropriate integer types to save the space. For example, if you just need to store the number of pages of a books, the SMALLINT is sufficient.

## Db2 integer example

First, [create a table](https://www.db2tutorial.com/db2-basics/db2-create-table/) named db2\_integers that consists of some integer columns:



|  |  |
| --- | --- |
| 1  2  3  4  5 | CREATE TABLE db2\_integers(      smallint\_col SMALLINT,      int\_col INT,      bigint\_col BIGINT  ); |

Second, [insert](https://www.db2tutorial.com/db2-basics/db2-insert/) some values into the table:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | INSERT INTO db2\_integers (      smallint\_col,      int\_col,      bigint\_col )  VALUES (      32767,      2147483647,      9223372036854775807  ); |

Third, query data from the db2\_integers table:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM db2\_integers; |

Here is the output:

Db2 Integer conversion example

Fourth, try to insert integers that exceed the range of the integer columns:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | INSERT INTO db2\_integers (      smallint\_col,      int\_col,      bigint\_col )  VALUES (      32768,      2147483648,      9223372036854775808  ); |

Db2 issued the following error message:



|  |  |
| --- | --- |
| 1 | SQL0413N  Overflow occurred during numeric data type conversion. |

Fifth, attempt to insert decimal numbers instead of integers into the integer columns:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | INSERT INTO db2\_integers (      smallint\_col,      int\_col,      bigint\_col )  VALUES (      32766.6,      2147483646.6,      9223372036854775806.6  ); |

Db2 implicitly converted the decimal numbers to integer numbers before insert:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM db2\_integers; |

The output is:

Db2 Integer example

# Db2 Decimal

**Summary**: in this tutorial, you’ll learn about Db2 decimal type and how to use it to store decimal numbers in tables.

### Introduction to Db2 decimal type

A decimal number consists of two parts: the whole part and the fractional part. The whole part and fractional part is separated by a decimal point (.)

In Db2, you declare a decimal number using one of the following forms:



|  |  |
| --- | --- |
| 1  2  3 | NUMERIC(p,s)  DECIMAL(p,s)  DEC(p,s) |

In these syntaxes:

* p is the precision which is the maximum number of decimal digits including both whole part and fractional part. e.g., 12.345 has a maximum precision of 5.
* s is called scale which is the number of decimal digits in the fractional part e.g., for the number 12.345, s is 3

So to store the number like 12.345 you need to declare the column that has one of the following syntaxes:



|  |  |
| --- | --- |
| 1  2  3 | NUMERIC(5,3)  DECIMAL(5,3)  DEC(5,3) |

The maximum precision of a decimal in Db2 is 31 digits. And decimal type has the maximum range from 1 – 10³¹ to 10³¹ – 1.

## Db2 decimal type example

First, [create a table](https://www.db2tutorial.com/db2-basics/db2-create-table/) named db2\_decimals that has a decimal column:



|  |  |
| --- | --- |
| 1  2  3 | CREATE TABLE db2\_decimals(      dec\_col NUMERIC(5,3)  ); |

Second, [insert](https://www.db2tutorial.com/db2-basics/db2-insert/) a decimal number into the table:



|  |  |
| --- | --- |
| 1  2 | INSERT INTO db2\_decimals(dec\_col)  VALUES(12.345); |

Third, [query](https://www.db2tutorial.com/db2-basics/db2-select/) data from the db2\_decimals table:



|  |  |
| --- | --- |
| 1  2 | SELECT dec\_col  FROM db2\_decimals; |

Here is the output:



|  |  |
| --- | --- |
| 1  2  3 | DEC\_COL  -------  12.345 |

Fourth, insert another decimal number into the db2\_decimals table:



|  |  |
| --- | --- |
| 1  2 | INSERT INTO db2\_decimals(dec\_col)  VALUES(12.3456); |

Db2 truncated the decimal number before insert. Here are the contents of the table:



|  |  |
| --- | --- |
| 1  2  3  4 | DEC\_COL  -------  12.345  12.345 |

Fifth, insert a decimal number that exceeds the range of the decimal column:



|  |  |
| --- | --- |
| 1  2 | INSERT INTO db2\_decimals(dec\_col)  VALUES(123.456); |

Db2 issued the following error:



|  |  |
| --- | --- |
| 1 | SQL0413N  Overflow occurred during numeric data type conversion. |

# Db2 VARCHAR

**Summary**: in this tutorial, you’ll learn how to use the Db2 VARCHAR data type to store variable-length character strings.

## Overview of Db2 VARCHAR type

Db2 VARCHAR type is used to store variable-length character strings. To define a variable-length character string column, you use the following syntax:



|  |  |
| --- | --- |
| 1 | column\_name VARCHAR(n) |

In this syntax, n is a positive integer that represents the maximum length of n bytes that the column can store. n also must be greater than zero and less than 32,740.

If you need to store a string whose length is longer than this, you should use the VARBINARY(n) data type instead.

## Db2 VARCHAR type examples

Let’s create a new table to demonstrate the characteristics of VARCHAR(n) data type.

This statement [creates a new table](https://www.db2tutorial.com/db2-basics/db2-create-table/) named db2\_varchars that has a VARCHAR(20) column:



|  |  |
| --- | --- |
| 1  2  3 | CREATE TABLE db2\_varchars (     v VARCHAR(20) NOT NULL  ); |

### 1) Insert a string into a variable-length character string column example

The following example inserts a string into the v column of the db2\_varchars table:



|  |  |
| --- | --- |
| 1  2  3 | INSERT INTO db2\_varchars (v)  VALUES     ('Db2 Tutorial'); |

It worked as expected because the string Db2 Tutorial has the length which is less than 20.

However, this statement attempts to insert a string whose length is 22 into the VARCHAR(20) column and fails:



|  |  |
| --- | --- |
| 1  2  3 | INSERT INTO db2\_varchars(v)  VALUES     ('A Guide to Db2 VARCHAR'); |

Db2 issued the following error message:



|  |  |
| --- | --- |
| 1 | SQL0433N  Value "A Guide to Db2 VARCHAR" is too long. |

### 2) Insert a Unicode string into a variable-length character string column example

This example inserts a Unicode string die Prüfung into the VARCHAR(20) column of the db2\_varchars table:



|  |  |
| --- | --- |
| 1  2  3 | INSERT INTO db2\_varchars(v)  VALUES     ('die Prüfung'); |

To view the data in the db2\_varchars table, you use the following SELECT statement:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM db2\_varchars; |

Here is the output:



|  |  |
| --- | --- |
| 1  2  3  4 | V  --------------------  Db2 Tutorial  die Prüfung |

# Db2 CHAR

**Summary**: in this tutorial, you will learn how to use Db2 CHAR data type to store fixed-length character string in the database.

## Introduction to Db2 CHAR data type

The CHARACTER or CHAR data type allows you to store fixed-length character strings in the database.

To declare a column of the CHAR type, you use the following syntax:



|  |  |
| --- | --- |
| 1 | CHAR(n) |

It is equivalent to the following syntax:



|  |  |
| --- | --- |
| 1 | CHARACTER(n) |

In this syntax, n is the maximum number of bytes that a CHAR column can store. The range of n is 1 to 255; it defaults to 1.

Notice that if you store Unicode characters,  one character may take more than one byte.

## Db2 CHAR type examples

Let’s [create a new table](https://www.db2tutorial.com/db2-basics/db2-create-table/) named db2\_characters to store the fixed length character strings:



|  |  |
| --- | --- |
| 1  2  3 | CREATE TABLE db2\_characters(      char\_col CHAR(3)  ); |

### 1) Insert a fixed-length character string into the CHAR column example

First, [insert](https://www.db2tutorial.com/db2-basics/db2-insert/) a string into the CHAR column:



|  |  |
| --- | --- |
| 1  2 | INSERT INTO db2\_characters(char\_col)  VALUES('abc'); |

Then, [query data](https://www.db2tutorial.com/db2-basics/db2-select/) from the db2\_characters table:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM db2\_characters; |

Here is the output:



|  |  |
| --- | --- |
| 1  2  3 | CHAR\_COL  --------  abc |

### 2) Padding space before inserting example

The following statement inserts the string xy whose length is 2 into the CHAR(3) column:



|  |  |
| --- | --- |
| 1  2 | INSERT INTO db2\_characters(char\_col)  VALUES('xy'); |

In this case, Db2 padded space to the string xy to make the length of the string 3 before inserting it into the CHAR(3) column.

To verify it, you can use the CHARACTER\_LENGTH() function to get the number of characters of each value in the char\_col column:



|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT      char\_col,      CHARACTER\_LENGTH(char\_col) length  FROM      db2\_characters; |

The following shows the output:



|  |  |
| --- | --- |
| 1  2  3  4 | CHAR\_COL      LENGTH  -------- -----------  abc                3  xy                 3 |

### 3) Inserting a Unicode character string into CHAR column example

The following statement inserts a Unicode character into the CHAR(3) column:



|  |  |
| --- | --- |
| 1  2 | INSERT INTO db2\_characters(char\_col)  VALUES('ü'); |

The character ü takes two bytes in UTF-8.



|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT     char\_col,     CHARACTER\_LENGTH(char\_col) length  FROM     db2\_characters; |

Here is the result set of the query:



|  |  |
| --- | --- |
| 1  2  3  4  5 | CHAR\_COL      LENGTH  -------- -----------  abc                3  xy                 3  ü                  2 |

# Db2 DATE Type

**Summary**: in this tutorial, you will learn how to use the Db2 DATE type and how to use it to store dates in the database.

## Introduction to the Db2 DATE type

In Db2, a date represents a point in time using the Gregorian calendar. A date consists of three parts: year, month, and day.

* The range of the year is from 0001 to 9999.
* The range of the month is 1 to 12
* The range of the day is 1 to 28, 29, 30 or 31, depending on the month and year.

In other words, the range of a date value is from 0001-01-01 to 9999-12-31.

The syntax for the type of a date value:



|  |  |
| --- | --- |
| 1 | DATE |

Internally, Db2 uses 4 bytes to represent a date value. The first two bytes represent the year, the third byte represents the month, and the last byte represents the day. Each byte stores two packed decimal digits.

Db2 stores date data in a special internal format. For displaying, Db2 converts the internal format to one of the following formats:

| **Format name** | **Abbreviation** | **Date format** | **Example** |
| --- | --- | --- | --- |
| International Standards Organization | ISO | yyyy-mm-dd | 2019-12-15 |
| IBM® USA standard | USA | mm/dd/yyyy | 12/15/2019 |
| IBM European standard | EUR | dd.mm.yyyy | 15.12.2019 |
| Japanese industrial standard Christian era | JIS | yyyy-mm-dd | 2019-12-15 |
| Installation-defined | LOCAL | Any installation-defined form |  |

## Date Literals

The following string represents a date value:



|  |  |
| --- | --- |
| 1 | '2019-05-06' |

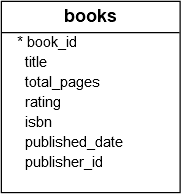
In this format, the year is 2019, the month is 05 (May), and the day is 06.

A date literal must conform to the following rules:

* A date literal cannot have leading blank but can have trailing blanks so ' 2019-05-06' is an invalid date while '2019-05-06 ' is a valid date.
* A date literal can omit the leading zero from the month and day of the date, for example, '2019-05-06','2019-05-6','2019-5-06', and '2019-05-06' represent the same date.
* A date literal must include leading zeros in the year element e.g., you cannot have a date literal like '80-10-20', it must be '0800-10-20'.

## Db2 DATE type examples

We’ll use the books table from the [sample database](https://www.db2tutorial.com/getting-started/db2-sample-database/) to demonstrate the date type.



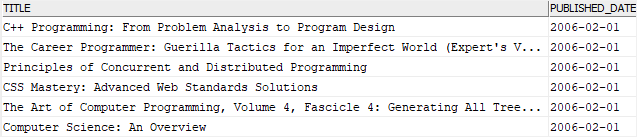
### 1) Using date literal examples

This example finds all books published on Feb 1st, 2006. It compares the published date with a date literal in the [WHERE](https://www.db2tutorial.com/db2-basics/db2-where/) clause:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT      title,      published\_date  FROM      books  WHERE      published\_date = '2006-02-01'; |

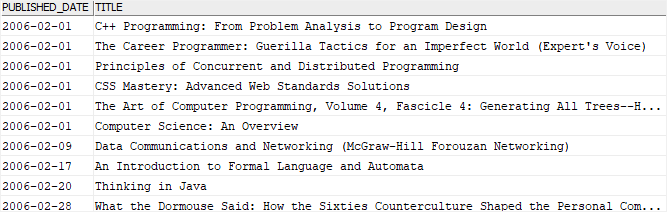
Here is the output:



The following statement finds all books published between Feb 1 2006 and Feb 28 2006:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | SELECT      published\_date,      title  FROM      books  WHERE      published\_date          BETWEEN '2006-02-01'              AND '2006-02-28'  ORDER BY      published\_date; |



### 2) Creating a table with a date column

First, [create a table](https://www.db2tutorial.com/db2-basics/db2-create-table/) to store the reading list of users:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | CREATE TABLE reading\_lists(     user\_id INT NOT NULL,      book\_id INT NOT NULL,      added\_on DATE DEFAULT CURRENT\_DATE,      PRIMARY KEY(user\_id, book\_id)  ); |

The reading\_lists table has the added\_on column which is a DATE column whose default value is the current date.

Next, [insert](https://www.db2tutorial.com/db2-basics/db2-insert/) a new row into the reading\_list table:



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO      reading\_lists(user\_id, book\_id)  VALUES      (1,1); |

Because we didn’t supply a value to the added\_on column, it took the default value which is the current date.

Then, query data from the reading\_lists table:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM reading\_lists; |

Db2 Date - default value for a date column

After that, insert a new row into the reading\_lists table, but provide the addon\_date column with a date literal:



|  |  |
| --- | --- |
| 1  2  3  4 | INSERT INTO      reading\_lists(user\_id, book\_id, added\_on)  VALUES      (2,2,'2019-5-3'); |

Note that we didn’t use the leading zero in the month and day element of the date literal.

Finally, verify the content of the reading\_lists table by using the following [SELECT](https://www.db2tutorial.com/db2-basics/db2-select/) statement:



|  |  |
| --- | --- |
| 1 | SELECT \* FROM reading\_lists; |

Db2 Date - insert date literal example

## Common Db2 date functions

Let’s quickly examine some common date functions.

### 1) Getting the current date

To get the current date of the operating system on which the Db2 instance is running, you use the CURRENT\_DATE function:



|  |  |
| --- | --- |
| 1  2  3  4 | SELECT      CURRENT\_DATE  FROM      sysibm.sysdummy1; |

Or you can use the following statement:



|  |  |
| --- | --- |
| 1  2  3  4 | SELECT      CURRENT DATE  FROM      sysibm.sysdummy1; |

Note that the sysdummy1 is a catalog table that consists of one row. The sysdummy1 resides in the sysibm schema. It is used in the SQL statement that requires a table reference.

### 2) Extracting the year, month, and day from a date

To extract the date parts such as the year, month, and day from a date, you use the YEAR(), MONTH(), and DAY() function respectively.



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT      YEAR(CURRENT\_DATE) current\_year,      MONTH(CURRENT\_DATE) current\_month,      DAY(CURRENT\_DATE) current\_day  FROM      sysibm.sysdummy1; |

### 3) Extracting the date from a timestamp

The extract the date from a timestamp, you use the DATE() function. The following example returns the date from the current timestamp:



|  |  |
| --- | --- |
| 1  2  3  4 | SELECT      DATE (CURRENT\_TIMESTAMP)  FROM      sysibm.sysdummy1; |

### 4) Formatting date in various formats

The CHAR() function formats a date in a specified format. For example, the following statement returns the book title and published date in ISO, USA, EUR, and JIS formats:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | SELECT      title,      CHAR(published\_date,ISO) published\_date\_iso,      CHAR(published\_date,USA) published\_date\_usa,      CHAR(published\_date,EUR) published\_date\_eur,      CHAR(published\_date,JIS) published\_date\_jis  FROM      books;  ORDER BY title; |

# Db2 COALESCE

**Summary**: in this tutorial, you will learn about the Db2 COALESCE() function and how to use it to substitute NULL values.

## Introduction to Db2 COALESCE() function

The basic syntax of the COALESCE() function is the following:



|  |  |
| --- | --- |
| 1 | COALESCE(v1,v2,...); |

The COALESCE() function accepts a number of arguments and returns the first non-NULL argument. If all arguments are NULL, the COALESCE() function returns NULL.

Here is a simple example of using the COALESCE() function:



|  |  |
| --- | --- |
| 1  2  3  4 | SELECT     COALESCE(NULL, 1, 2) result  FROM     SYSIBM.SYSDUMMY1; |

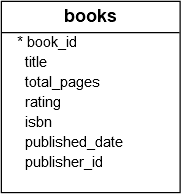
The output is as follows:



|  |  |
| --- | --- |
| 1  2  3 | RESULT  -----------  1 |

## Db2 COALESCE() function examples

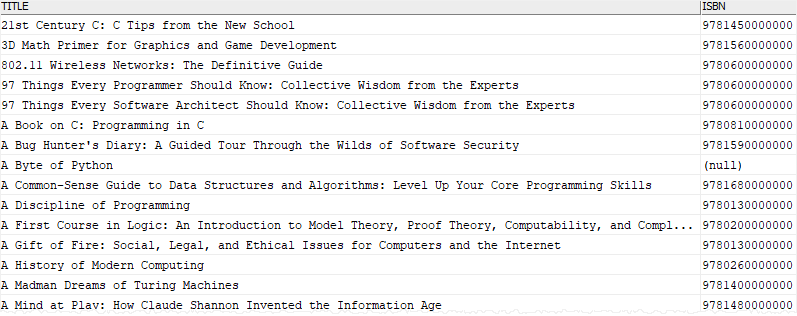
We’ll use the books table from the [sample database](https://www.db2tutorial.com/getting-started/db2-sample-database/) to demonstrate the COALESCE() function.



This query returns the book title and ISBN of all books from the books table:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT     title,     isbn  FROM     books  ORDER BY title; |

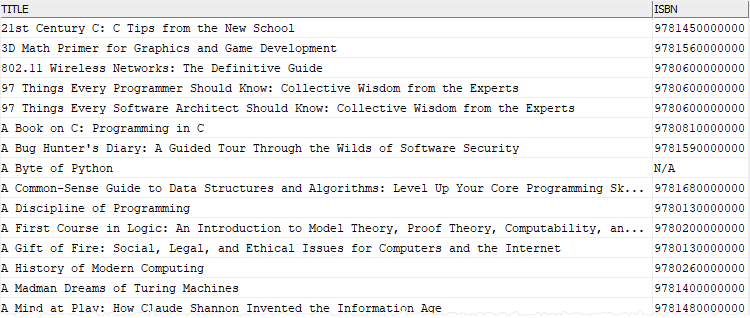


The ISBN column has NULL values. To substitute these NULL values by more meaningful values, you can use the COALESCE() function as shown in the following query:



|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT      title,     COALESCE(isbn, 'N/A') isbn  FROM      books  ORDER BY title; |

Here is the output:



In this example, if the value in the isbn column is NULL, the COALESCE function will substitute it by the N/A string. Otherwise, it returns the value of the isbn column.

## Db2 COALESCE() function and CASE expression

This COALESCE() function:



|  |  |
| --- | --- |
| 1 | COALESCE(e1,e2,e3) |

has the same effect as the following searched [CASE](https://www.db2tutorial.com/db2-basics/db2-case-expression/) expression:



|  |  |
| --- | --- |
| 1  2  3  4  5 | CASE     WHEN e1 IS NOT NULL THEN e1     WHEN e2 IS NOT NULL THEN e2     ELSE e3  END |

For example, this query uses the CASE expression to return the same result set as the example above:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | SELECT      title,      (CASE          WHEN isbn IS NULL THEN 'N/A'          ELSE isbn      END) AS isbn  FROM      books  ORDER BY     title; |