|  |
| --- |
| **What is Database**  A database is a collection of information that is organized so that it can easily be accessed, managed, and updated.  A **database** is an organized collection of data. It is the collection of [schemas](https://en.wikipedia.org/wiki/Database_schema), [tables](https://en.wikipedia.org/wiki/Table_(database)), [queries](https://en.wikipedia.org/wiki/Query_language), reports, [views](https://en.wikipedia.org/wiki/View_(SQL)), and other objects. The data are typically organized to model aspects of reality in a way that supports [processes](https://en.wikipedia.org/wiki/Process_(computing)) requiring information, such as modelling the availability of rooms in hotels in a way that supports finding a hotel with vacancies. |
| **What is Table**  In relational databases and flat file databases, a table is a set of data elements (values) using a model of vertical columns (identifiable by name) and horizontal rows, the cell being the unit where a row and column intersect. A table has a specified number of columns, but can have any number of rows.  a table (sometimes called a [file](http://searchexchange.techtarget.com/definition/file)) organizes the information about a single topic into rows and columns. For example, a database for a business would typically contain a table for customer information, which would store customers' account numbers, addresses, phone numbers, and so on as a series of columns. Each single piece of data (such as the account number) is a [field](http://searchoracle.techtarget.com/definition/field) in the table. A column consists of all the entries in a single field, such as the telephone numbers of all the customers. Fields, in turn, are organized as [record](http://searchoracle.techtarget.com/definition/record)s, which are complete sets of information (such as the set of information about a particular customer), each of which comprises a row. The process of [normalization](http://searchsqlserver.techtarget.com/definition/normalization) determines how data will be most effectively organized into tables. |
| **What is Column**  In the context of a relational **database**, a **column** is a set of data values of a particular simple type, one for each row of the table. The **columns** provide the structure according to which the rows are composed.  The columns in a table are the set of facts that we keep track of about that type of object. A column is also called an *attribute*. |
| **What is Row**  n the context of a relational **database**, a **row**—also called a record or tuple—represents a single, implicitly structured data item in a table. In simple terms, a **database** table can be thought of as consisting of **rows** and columns or fields.  Each row in a database table represents one instance of the type of object described in that table. A row is also called a *record*. |
| A SQL join is a Structured Query Language (**SQL**) instruction to combine data from two sets of data (e.g. two tables). Before we dive into the details of a SQL join, let’s briefly discuss what SQL is, and why someone would want to perform a SQL join. |
| **Example for Inner join**  The INNER JOIN keyword selects all rows from both tables as long as there is a match between the columns in both tables.  ***SQL INNER JOIN Syntax***  SELECT *column\_name(s)* FROM *table1* INNER JOIN *table2* ON *table1.column\_name*=*table2.column\_name*;  or:  SELECT *column\_name(s)* FROM *table1* JOIN *table2* ON *table1.column\_name*=*table2.column\_name*;  **PS!** INNER JOIN is the same as JOIN.  QL INNER JOIN  The following SQL statement will return all customers with orders:  Example  SELECT Customers.CustomerName,Orders.OrderID FROM Customers INNER JOIN Orders ON Customers.CustomerID=Orders.CustomerID ORDER BY Customers.CustomerName; |
| **Example for Left outer join**  The LEFT JOIN keyword returns all rows from the left table (table1), with the matching rows in the right table (table2). The result is NULL in the right side when there is no match.  ***Syntax***  SELECT *column\_name(s)* FROM *table1* LEFT JOIN *table2* ON *table1.column\_name*=*table2.column\_name*;  or:  SELECT *column\_name(s)* FROM *table1* LEFT OUTER JOIN *table2* ON *table1.column\_name*=*table2.column\_name*;  In some databases LEFT JOIN is called LEFT OUTER JOIN.  QL LEFT JOIN  The following SQL statement will return all customers, and any orders they might have:  Example  SELECT Customers.CustomerName,Orders.OrderID FROM Customers LEFT JOIN Orders ON Customers.CustomerID=Orders.CustomerID ORDER BY Customers.CustomerName; |
| **Example for Right outer join**  The RIGHT JOIN keyword returns all rows from the right table (table2), with the matching rows in the left table (table1). The result is NULL in the left side when there is no match.  ***Syntax***  SELECT *column\_name(s)* FROM *table1* RIGHT JOIN *table2* ON *table1.column\_name*=*table2.column\_name*;  or:  SELECT *column\_name(s)* FROM *table1* RIGHT OUTER JOIN *table2* ON *table1.column\_name*=*table2.column\_name*;  **I**n some databases RIGHT JOIN is called RIGHT OUTER JOIN.  QL RIGHT JOIN |
| **Example for Max, sum, Avg**  The SUM function is an aggregate function that adds up all values in a specific column. You can only use the SUM function with numeric values either integers or decimals.  SELECT SUM(Total) FROM Orders WHERE OrderDate BETWEEN ‘3/1'2014' AND ‘3/31/2014'  The AVG function works in a similar way as SUM. The difference is that the AVG function adds up or sums up all values and then calculates the average. The average is based on the number of records returned by the SQL statement, so you receive different results based on your WHERE clause.  SELECT AVG(Total) FROM Orders WHERE OrderDate BETWEEN ‘3/1'2014' AND ‘3/31/2014'  The MIN and MAX functions find the minimum or maximum value in a record set.  SELECT MAX(Total) FROM Order |
| **Example for Group by**  The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns.  ***Syntax***  SELECT column\_name,aggregate\_function(column\_name) FROM table\_name WHERE column\_name operatorvalue GROUP BY column\_name;  Example  SELECT Shippers.ShipperName,COUNT(Orders.OrderID) AS NumberOfOrders  FROM Orders LEFT JOIN Shippers ON Orders.ShipperID=Shippers.ShipperID GROUP BY ShipperName; |
| **Example for Having**  Syntax  SELECT column\_name,aggregate\_function(column\_name) FROM table\_name WHERE column\_name operatorvalue GROUP BY column\_name HAVING aggregate\_function(column\_name) operator value;  Example  SELECT Employees.LastName, COUNT(Orders.OrderID) AS NumberOfOrders FROM (Orders INNER JOIN Employees ON Orders.EmployeeID=Employees.EmployeeID) GROUP BY LastName HAVING COUNT(Orders.OrderID) > 10; |
| **Example for Where condition**  The WHERE clause is used to extract only those records that fulfill a specified criterion.  ***Syntax***  SELECT *column\_name*,*column\_name* FROM *table\_name* WHERE *column\_name operator value*;  SELECT \* FROM Customers WHERE Country='Mexico'; |
| **Example for Primary key**  The PRIMARY KEY constraint uniquely identifies each record in a database table.  Primary keys must contain UNIQUE values.  A primary key column cannot contain NULL values.  Most tables should have a primary key, and each table can have only ONE primary key.  The following SQL creates a PRIMARY KEY on the "P\_Id" column when the "Persons" table is created:  **MySQL:**  CREATE TABLE Persons ( P\_Id int NOT NULL, LastName varchar(255) NOT NULL, FirstName varchar(255), Address varchar(255), City varchar(255), PRIMARY KEY (P\_Id) ) |
| **Example for Foreign key**  A FOREIGN KEY in one table points to a PRIMARY KEY in another table.  The following SQL creates a FOREIGN KEY on the "P\_Id" column when the "Orders" table is created:  **MySQL:**  CREATE TABLE Orders ( O\_Id int NOT NULL, OrderNo int NOT NULL, P\_Id int, PRIMARY KEY (O\_Id), FOREIGN KEY (P\_Id) REFERENCES Persons(P\_Id) ) |
|  |
| Finding second highest salary from row table |
| (SELECT MAX(Salary) FROM Employee WHERE Salary NOT IN (SELECT MAX(Salary) FROM Employee )) as [2nd\_max\_salary] |
| **Need to practice:** |
| Tables for employee management system |
|  |
| Tables for e-commerce management system |
|  |
| Tables for Library management system |

Check the database for orders and den see the cart.