**There are different types of relations in the database:**

**One-to-One** – This is a connection between two tables in which each record in one table corresponds to the maximum of one record in the other.

**One-to-Many and Many-to-One** – This is the most frequent connection, in which a record in one table is linked to several records in another.

**Many-to-Many** – This is used when defining a relationship that requires several instances on each sides.

**Self-Referencing Relationships** – When a table has to declare a connection with itself, this is the method to employ.

**What is Normalization and what are the advantages of it?**

[Normalization in SQL](https://www.edureka.co/blog/normalization-in-sql/) is the process of organizing data to avoid duplication and redundancy. Some of the advantages are:

* Better Database organization
* More Tables with smaller rows
* Efficient data access
* Greater Flexibility for Queries
* Quickly find the information
* Easier to implement Security
* Allows easy modification
* Reduction of redundant and duplicate data
* More Compact Database
* Ensure Consistent data after modification

Normalization in SQL | 1NF, 2NF, 3NF and BCNF in Database | Edureka

This article on Normalization in SQL will help you to learn how to reduce the redundancy of data and decrease the anomalies of the database.

**1. What is Database?**

A database is an organized collection of data, stored and retrieved digitally from a remote or local computer system. Databases can be vast and complex, and such databases are developed using fixed design and modeling approaches.

**2. What is DBMS?**

DBMS stands for Database Management System. DBMS is a system software responsible for the creation, retrieval, updation, and management of the database. It ensures that our data is consistent, organized, and is easily accessible by serving as an interface between the database and its end-users or application software.

**3. What is RDBMS? How is it different from DBMS?**

RDBMS stands for Relational Database Management System. The key difference [here](https://www.interviewbit.com/blog/difference-between-dbms-and-rdbms/), compared to DBMS, is that RDBMS stores data in the form of a collection of tables, and relations can be defined between the common fields of these tables. Most modern database management systems like MySQL, Microsoft SQL Server, Oracle, IBM DB2, and Amazon Redshift are based on RDBMS.

Difference Between DBMS and RDBMS - DBMS Vs RDBMS

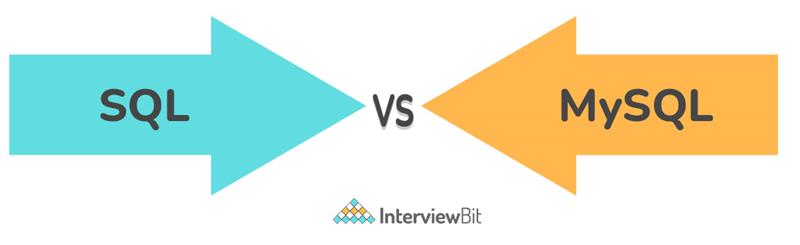
Table Of Contents show What is DBMS? What is RDBMS? Difference Between DBMS and RDBMS Conclusion FAQs Additional Resources Imagine a time when vital data was documented using pen and paper?

**What is SQL?**

SQL stands for Structured Query Language. It is the standard language for relational database management systems. It is especially useful in handling organized data comprised of entities (variables) and relations between different entities of the data.

**5. What is the difference between SQL and MySQL?**

SQL is a standard language for retrieving and manipulating structured databases. On the contrary, MySQL is a relational database management system, like SQL Server, Oracle or IBM DB2, that is used to manage SQL databases.



**6. What are Tables and Fields?**

A table is an organized collection of data stored in the form of rows and columns. Columns can be categorized as vertical and rows as horizontal. The columns in a table are called fields while the rows can be referred to as records.

**7. What are Constraints in SQL?**

Constraints are used to specify the rules concerning data in the table. It can be applied for single or multiple fields in an SQL table during the creation of the table or after creating using the ALTER TABLE command. The constraints are:

* **NOT NULL** - Restricts NULL value from being inserted into a column.
* **CHECK** - Verifies that all values in a field satisfy a condition.
* **DEFAULT** - Automatically assigns a default value if no value has been specified for the field.
* **UNIQUE** - Ensures unique values to be inserted into the field.
* **INDEX** - Indexes a field providing faster retrieval of records.
* **PRIMARY KEY** - Uniquely identifies each record in a table.
* **FOREIGN KEY** - Ensures referential integrity for a record in another table.

**8. What is a Primary Key?**

The PRIMARY KEY constraint uniquely identifies each row in a table. It must contain UNIQUE values and has an implicit NOT NULL constraint.  
A table in SQL is strictly restricted to have one and only one primary key, which is comprised of single or multiple fields (columns).

**What is a UNIQUE constraint?**

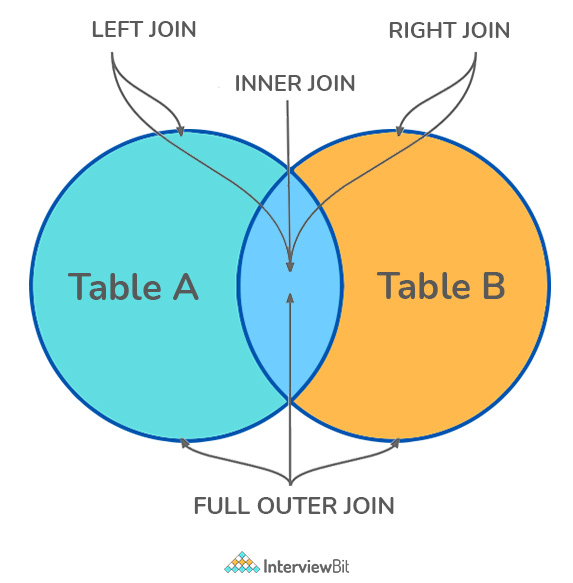
A UNIQUE constraint ensures that all values in a column are different. This provides uniqueness for the column(s) and helps identify each row uniquely. Unlike primary key, there can be multiple unique constraints defined per table. The code syntax for UNIQUE is quite similar to that of PRIMARY KEY and can be used interchangeably.

**What is a Foreign Key?**

A FOREIGN KEY comprises of single or collection of fields in a table that essentially refers to the PRIMARY KEY in another table. Foreign key constraint ensures referential integrity in the relation between two tables.  
The table with the foreign key constraint is labeled as the child table, and the table containing the candidate key is labeled as the referenced or parent table.

**What is a Join? List its different types.**

The [**SQL Join**](https://www.scaler.com/topics/joins-in-sql/) clause is used to combine records (rows) from two or more tables in a SQL database based on a related column between the two.



There are four different types of JOINs in SQL:

* **(INNER) JOIN:** Retrieves records that have matching values in both tables involved in the join. This is the widely used join for queries.

SELECT \*

FROM Table\_A

JOIN Table\_B;

SELECT \*

FROM Table\_A

INNER JOIN Table\_B;

* **LEFT (OUTER) JOIN:** Retrieves all the records/rows from the left and the matched records/rows from the right table.

SELECT \*

FROM Table\_A A

LEFT JOIN Table\_B B

ON A.col = B.col;

* **RIGHT (OUTER) JOIN:** Retrieves all the records/rows from the right and the matched records/rows from the left table.

SELECT \*

FROM Table\_A A

RIGHT JOIN Table\_B B

ON A.col = B.col;

* **FULL (OUTER) JOIN:** Retrieves all the records where there is a match in either the left or right table.

SELECT \*

FROM Table\_A A

FULL JOIN Table\_B B

ON A.col = B.col;

Joins in SQL | SQL Joins - Inner, Left, Right and Full Join - Scaler Topics

SQL Joins are mostly used when a user is trying to extricate data from multiple tables simultaneously. In this article, learn different types of Join in SQL by Scaler Topics.

**What is a Self-Join?**

A **self JOIN** is a case of regular join where a table is joined to itself based on some relation between its own column(s). Self-join uses the INNER JOIN or LEFT JOIN clause and a table alias is used to assign different names to the table within the query.

**What is a Query?**

A query is a request for data or information from a database table or combination of tables. A database query can be either a select query or an action query.

**What is a Subquery? What are its types?**

A subquery is a query within another query, also known as a **nested query** or **inner query**. It is used to restrict or enhance the data to be queried by the main query, thus restricting or enhancing the output of the main query respectively. For example, here we fetch the contact information for students who have enrolled for the maths subject:

**What is the SELECT statement?**

SELECT operator in SQL is used to select data from a database. The data returned is stored in a result table, called the result-set.

**What are some common clauses used with SELECT query in SQL?**

Some common SQL clauses used in conjuction with a SELECT query are as follows:

* **WHERE** clause in SQL is used to filter records that are necessary, based on specific conditions.
* **ORDER BY** clause in SQL is used to sort the records based on some field(s) in ascending (**ASC**) or descending order (**DESC)**.

SELECT \*

FROM myDB.students

WHERE graduation\_year = 2019

ORDER BY studentID DESC;

* **GROUP BY** clause in SQL is used to group records with identical data and can be used in conjunction with some aggregation functions to produce summarized results from the database.
* **HAVING** clause in SQL is used to filter records in combination with the GROUP BY clause. It is different from WHERE, since the WHERE clause cannot filter aggregated records.

**What are UNION, MINUS and INTERSECT commands?**

The **UNION** operator combines and returns the result-set retrieved by two or more SELECT statements.  
The **MINUS** operator in SQL is used to remove duplicates from the result-set obtained by the second SELECT query from the result-set obtained by the first SELECT query and then return the filtered results from the first.  
The **INTERSECT** clause in SQL combines the result-set fetched by the two SELECT statements where records from one match the other and then returns this intersection of result-sets.

Certain conditions need to be met before executing either of the above statements in SQL -

* Each SELECT statement within the clause must have the same number of columns
* The columns must also have similar data types
* The columns in each SELECT statement should necessarily have the same order

SELECT name FROM Students /\* Fetch the union of queries \*/

UNION

SELECT name FROM Contacts;

SELECT name FROM Students /\* Fetch the union of queries with duplicates\*/

UNION ALL

SELECT name FROM Contacts;

SELECT name FROM Students /\* Fetch names from students \*/

MINUS /\* that aren't present in contacts \*/

SELECT name FROM Contacts;

SELECT name FROM Students /\* Fetch names from students \*/

INTERSECT /\* that are present in contacts as well \*/

SELECT name FROM Contacts;

**Write a SQL query to fetch "names" that are present in either table "accounts" or in table "registry".**

**Write a SQL query to fetch "names" that are present in "accounts" but not in table "registry".**

**Write a SQL query to fetch "names" from table "contacts" that are neither present in "accounts.name**

**List the different types of relationships in SQL.**

* **One-to-One** - This can be defined as the relationship between two tables where each record in one table is associated with the maximum of one record in the other table.
* **One-to-Many & Many-to-One** - This is the most commonly used relationship where a record in a table is associated with multiple records in the other table.
* **Many-to-Many** - This is used in cases when multiple instances on both sides are needed for defining a relationship.
* **Self-Referencing Relationships** - This is used when a table needs to define a relationship with itsel

**What is an Alias in SQL?**

An alias is a feature of SQL that is supported by most, if not all, RDBMSs. It is a temporary name assigned to the table or table column for the purpose of a particular SQL query. In addition, aliasing can be employed as an obfuscation technique to secure the real names of database fields. A table alias is also called a correlation name.

**What is Normalization?**

Normalization represents the way of organizing structured data in the database efficiently. It includes the creation of tables, establishing relationships between them, and defining rules for those relationships. Inconsistency and redundancy can be kept in check based on these rules, hence, adding flexibility to the database

**What are the TRUNCATE, DELETE and DROP statements?**

**DELETE** statement is used to delete rows from a table.

DELETE FROM Candidates

WHERE CandidateId > 1000;

**TRUNCATE** command is used to delete all the rows from the table and free the space containing the table.

TRUNCATE TABLE Candidates;

**DROP** command is used to remove an object from the database. If you drop a table, all the rows in the table are deleted and the table structure is removed from the database.

DROP TABLE Candidates;

**Write a SQL statement to wipe a table 'Temporary' from memory.**

**Write a SQL query to remove first 1000 records from table 'Temporary' based on 'id'.**

**Write a SQL statement to delete the table 'Temporary' while keeping its relations intact**

**What is the difference between DROP and TRUNCATE statements?**

If a table is dropped, all things associated with the tables are dropped as well. This includes - the relationships defined on the table with other tables, the integrity checks and constraints, access privileges and other grants that the table has. To create and use the table again in its original form, all these relations, checks, constraints, privileges and relationships need to be redefined. However, if a table is truncated, none of the above problems exist and the table retains its original structure.

**32. What is the difference between DELETE and TRUNCATE statements?**

The **TRUNCATE** command is used to delete all the rows from the table and free the space containing the table.  
The **DELETE** command deletes only the rows from the table based on the condition given in the where clause or deletes all the rows from the table if no condition is specified. But it does not free the space containing the table

**What are Aggregate and Scalar functions?**

An aggregate function performs operations on a collection of values to return a single scalar value. Aggregate functions are often used with the GROUP BY and HAVING clauses of the SELECT statement. Following are the widely used SQL aggregate functions:

* **AVG()** - Calculates the mean of a collection of values.
* **COUNT()** - Counts the total number of records in a specific table or view.
* **MIN()** - Calculates the minimum of a collection of values.
* **MAX()** - Calculates the maximum of a collection of values.
* **SUM()** - Calculates the sum of a collection of values.
* **FIRST()** - Fetches the first element in a collection of values.
* **LAST()** - Fetches the last element in a collection of values.

**Note:** All aggregate functions described above ignore NULL values except for the COUNT function.

A scalar function returns a single value based on the input value. Following are the widely used SQL scalar functions:

* **LEN()** - Calculates the total length of the given field (column).
* **UCASE()** - Converts a collection of string values to uppercase characters.
* **LCASE()** - Converts a collection of string values to lowercase characters.
* **MID()** - Extracts substrings from a collection of string values in a table.
* **CONCAT()** - Concatenates two or more strings.
* **RAND()** - Generates a random collection of numbers of a given length.
* **ROUND()** - Calculates the round-off integer value for a numeric field (or decimal point values).
* **NOW()** - Returns the current date & time.
* **FORMAT()** - Sets the format to display a collection of values

**What is Pattern Matching in SQL?**

SQL pattern matching provides for pattern search in data if you have no clue as to what that word should be. This kind of SQL query uses wildcards to match a string pattern, rather than writing the exact word. The LIKE operator is used in conjunction with **SQL Wildcards** to fetch the required information.

* **Using the % wildcard to perform a simple search**

The % wildcard matches zero or more characters of any type and can be used to define wildcards both before and after the pattern. Search a student in your database with first name beginning with the letter K:

SELECT \*

FROM students

WHERE first\_name LIKE 'K%'

**What is the difference between a primary key and a unique key?**

The primary key and unique key both are essential constraints of the SQL. The main difference among them is that the primary key identifies each record in the table. In contrast, the unique key prevents duplicate entries in a column except for a NULL value. The following comparison chart explains it more clearly:

**What are the different types of SQL operators?**

Operators are the special keywords or special characters reserved for performing particular operations. They are also used in SQL queries. We can primarily use these operators within the WHERE clause of SQL commands. It's a part of the command to filters data based on the specified condition. The SQL operators can be categorized into the following types:

* **Arithmetic operators:** These operators are used to perform mathematical operations on numerical data. The categories of this operators are addition (+), subtraction (-), multiplication (\*), division (/), remainder/modulus (%), etc.
* **Logical operators:** These operators evaluate the expressions and return their results in True or False. This operator includes ALL, AND, ANY, ISNULL, EXISTS, BETWEEN, IN, LIKE, NOT, OR, UNIQUE.
* **Comparison operators:** These operators are used to perform comparisons of two values and check whether they are the same or not. It includes equal to (=), not equal to (!= or <>), less than (<), greater than (>), less than or equal to (<=), greater than or equal to (>=), not less than (!<), not greater than (!>), etc.
* **Bitwise operators:** It is used to do bit manipulations between two expressions of integer type. It first performs conversion of integers into binary bits and then applied operators such as AND (& symbol), OR (|, ^), NOT (~), etc.
* **Compound operators:** These operators perform operations on a variable before setting the variable's result to the operation's result. It includes Add equals (+=), subtract equals (-=), multiply equals (\*=), divide equals (/=), modulo equals (%=), etc.
* **String operators:** These operators are primarily used to perform concatenation and pattern matching of strings. It includes + (String concatenation), += (String concatenation assignment),

**How do we use the DISTINCT statement? What is its use?**

The DISTINCT keyword is used to ensure that the fetched value always has unique values. It does not allow to have duplicate values. The DISTINCT keyword is used with the SELECT statement and retrieves different values from the table's column.

1. SELECT subject\_code, AVG (marks)
2. FROM Students
3. WHERE AVG(marks) > 70
4. GROUP BY subject\_code;

**What is SQL Injection?**

SQL injection is a type of vulnerability in website and web app code that allows attackers to control back-end operations and access, retrieve, and destroy sensitive data from databases.