Module 4 – Introduction to DBMS Introduction to SQL

1. What is SQL, and why is it essential in database management?

Structured Query Language (SQL) is the standard language used to interact with relational databases. It allows users to store, retrieve, update, and manage data efficiently through simple commands. Known for its user-friendly syntax and powerful capabilities, SQL is widely used across industries and applications. SQL is widely adopted due to its ease of use, efficiency, and compatibility across different database systems.

2. Explain the difference between DBMS and RDBMS.

Database management system, as the name suggests, is a management system that is used to manage the entire flow of data, i.e, the insertion of data or the retrieval of data, how the data is inserted into the database, or how fast the data should be retrieved, so DBMS takes care of all these features, as it maintains the uniformity of the database as well does the faster insertions as well as retrievals.

RDBMS on the other hand is a type of DBMS, as the name suggests it deals with relations as well as various key constraints. So here we have tables which are called schema and we have rows which are called tuples. It also aids in the reduction of data redundancy and the preservation of database integrity.

3. Describe the role of SQL in managing relational databases.

Structured Query Language (SQL) serves as the standard language for managing and interacting with relational databases. Its role encompasses various critical functions for defining, manipulating, querying, and controlling data within these systems.

4. What are the key features of SQL?

* **Data Definition Language (DDL):**

SQL provides commands to define and modify the structure of a database. This includes creating, altering, and dropping databases, tables, views, and indexes.

* **Data Manipulation Language (DML):**

SQL offers commands for manipulating data within a database. This encompasses inserting new records, updating existing records, and deleting records from tables.

* **Data Query Language (DQL):**

SQL is primarily known for its powerful querying capabilities. The SELECT statement allows users to retrieve data based on various criteria, including filtering, sorting, grouping, and joining data from multiple tables.

Practical

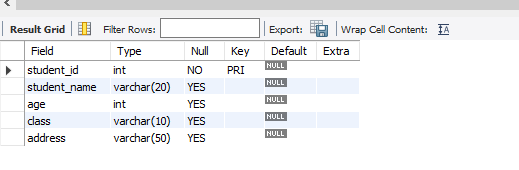
Lab 1: Create a new database named school\_db and a table called students with the following columns: student\_id, student\_name, age, class, and address.

create database student\_db;

use student\_db;

create table students(student\_id int primary key, student\_name varchar(20), age int, class varchar(10), address varchar(50));

desc students;



Lab 2: Insert five records into the students table and retrieve all records using the SELECT statement.

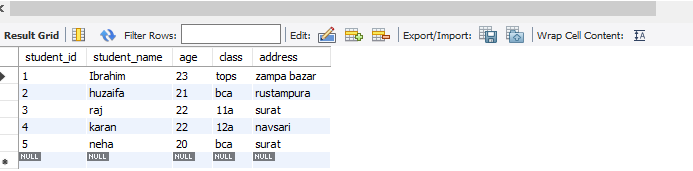
insert into students(student\_id, student\_name, age, class, address) value(01,"Ibrahim", 23,"tops", 'zampa bazar');

insert into students(student\_id, student\_name, age, class, address) value(02,'huzaifa', 21, 'bca', 'rustampura');

insert into students(student\_id, student\_name, age, class, address) value(03, 'raj', 22, '11a', 'surat');

insert into students(student\_id, student\_name, age, class, address) value(04, 'karan', 22, '12a', 'navsari');

insert into students(student\_id, student\_name, age, class, address) value(05, 'neha', 20, 'bca', 'surat');



2. SQL Syntax

1. What are the basic components of SQL syntax?

CREATE TABLE Employees (

EmployeeID INT PRIMARY KEY,

FirstName VARCHAR(50),

LastName VARCHAR(50)

);

ALTER TABLE Employees ADD COLUMN Email VARCHAR(100);

DROP TABLE Employees;

INSERT INTO Employees (EmployeeID, FirstName, LastName) VALUES (1, 'John', 'Doe');

UPDATE Employees SET Email = 'john.doe@example.com' WHERE EmployeeID = 1;

2. Write the general structure of an SQL SELECT statement.

SELECT *column1*,*column2, ...*  
FROM *table\_name*;

SELECT \* FROM Customers;

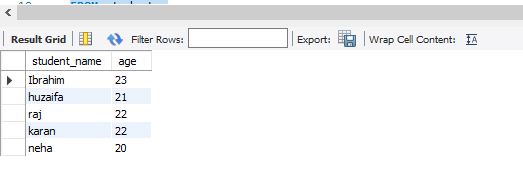
3. Explain the role of clauses in SQL statements.

|  |  |
| --- | --- |
| **WHERE** | The WHERE clause is used to filter records based on specific conditions. It is typically used in SELECT, UPDATE, and DELETE queries to restrict the data that is affected by these statements. For example, retrieving all employees with a salary above 50,000. |
| **ORDER BY** | The ORDER BY clause is used to sort the query results in either ascending or descending order. It is commonly used with numeric, date, and text fields to organize data meaningfully, such as sorting employees by their joining date. |
| **GROUP BY** | The GROUP BY clause groups records with the same values in specified columns and is used with aggregate functions like COUNT(), SUM(), AVG(), etc. For example, calculating total sales per region. |
| **HAVING** | The HAVING clause is similar to WHERE but is used to filter grouped records. It is used with GROUP BY to apply conditions on aggregated results, such as filtering groups where the total revenue exceeds a certain amount. |
| **LIMIT** | The LIMIT clause restricts the number of rows returned in a query result. This is especially useful in large databases where retrieving all records could be inefficient. For example, fetching the top 5 highest-paid employees. |
| **TOP** | The TOP clause, similar to LIMIT, is used in SQL Server to limit the number of rows returned. It helps in retrieving a specific subset of records efficiently. |
| **LIKE** | The LIKE clause filters results using pattern matching with wildcards (% for multiple characters and \_ for a single character). It is useful for searching partial matches in text fields, such as finding all customers whose names start with 'J'. |
| **FROM** | The FROM clause specifies the database table from which records will be retrieved. It is a fundamental part of SQL queries as it defines the source of data for SELECT, DELETE, and UPDATE statements. |
| **AND** | The AND clause is used to combine multiple conditions in a query, ensuring that all conditions must be met. It is useful in complex filtering scenarios, such as retrieving employees who work in a specific department and have a salary above 60,000. |
| **OR** | The OR clause is used to combine multiple conditions where at least one must be true. It is useful when searching for multiple criteria, such as retrieving customers from either New York or Los Angeles. |

Lab 1: Write SQL queries to retrieve specific columns (student\_name and age) from the students table.

SELECT student\_name, age

FROM students;

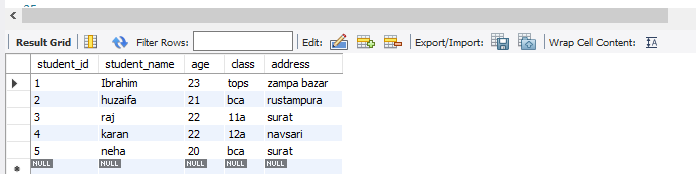


Lab 2: Write SQL queries to retrieve all students whose age is greater than 10.

SELECT \*

FROM students

WHERE age > 10;



3. SQL Constraints

1. What are constraints in SQL? List and explain the different types of constraints.

In SQL, constraints are rules applied to columns or tables in a relational database to limit the type of data that can be inserted, updated, or deleted. They ensure data integrity, consistency, and adherence to defined business rules or database requirements.

* **NOT NULL**: This constraint ensures that a column cannot store NULL values. Every row in that column must contain a value.

CREATE TABLE Employees (

EmployeeID INT NOT NULL,

FirstName VARCHAR(50) NOT NULL

);

* **UNIQUE**: This constraint ensures that all values in a column or a set of columns are unique. While multiple columns can have a UNIQUE constraint, the combination of values across those columns must be distinct.

CREATE TABLE Products (

ProductID INT UNIQUE,

ProductName VARCHAR(100) UNIQUE

);

* **PRIMARY KEY**: This constraint uniquely identifies each record in a table. It is a combination of NOT NULL and UNIQUE. A table can have only one primary key, which can consist of one or more columns.

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

CustomerName VARCHAR(100)

);

* **FOREIGN KEY**: This constraint establishes a link between two tables, ensuring referential integrity. It creates a relationship where values in a column (or set of columns) in one table (the referencing table) must match values in the primary key or a unique key of another table (the referenced table).

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

2.How do PRIMARY KEY and FOREIGN KEY constraints differ?

A [primary key](https://www.geeksforgeeks.org/sql/primary-key-constraint-in-sql/)is used to ensure that data in the specific column is unique. A column cannot have NULL values. It is either an existing table column or a column that is specifically generated by the database according to a defined sequence.

A [foreign key](https://www.geeksforgeeks.org/sql/foreign-key-constraint-in-sql/)is a column or group of columns in a [relational database](https://www.geeksforgeeks.org/dbms/relational-model-in-dbms/)table that provides a link between data in two tables. It is a column (or columns) that references a column (most often the primary key) of another table.

3. What is the role of NOT NULL and UNIQUE constraints?

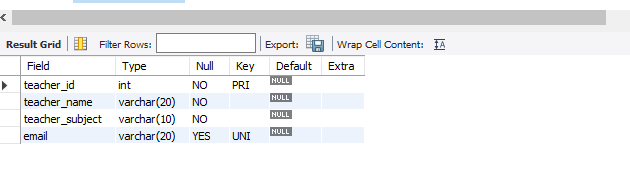
Ensures that a column cannot contain a NULL value. A NULL value indicates that the data is missing, unknown, or not applicable. By enforcing NOT NULL, you guarantee that every row in the table has a value for that specific column.

Ensures that all values in a column (or a set of columns) are unique within the table. No two rows can have the same value in that column.

Lab 1: Create a table teachers with the following columns: teacher\_id (Primary Key), teacher\_name (NOT NULL), subject (NOT NULL), and email (UNIQUE).

create table teachers(teacher\_id int primary key, teacher\_name varchar(20) not null, teacher\_subject varchar(10) not null, email varchar(20) unique );

desc teachers;



Lab 2: Implement a FOREIGN KEY constraint to relate the teacher\_id from the teachers table with the students table.

INSERT INTO teachers (teacher\_id, teacher\_name,teacher\_subject,email) values(1, 'neha', 'html', 'neha@gmail.com');

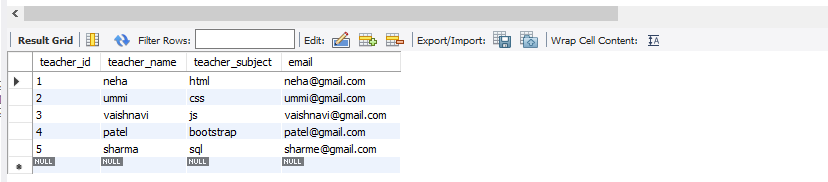
INSERT INTO teachers (teacher\_id, teacher\_name,teacher\_subject,email) values(2, 'ummi', 'css', 'ummi@gmail.com');

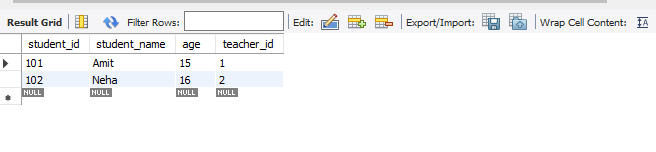
INSERT INTO teachers (teacher\_id, teacher\_name,teacher\_subject,email) values(3, 'vaishnavi', 'js', 'vaishnavi@gmail.com');

INSERT INTO teachers (teacher\_id, teacher\_name,teacher\_subject,email) values(4, 'patel', 'bootstrap', 'patel@gmail.com');

INSERT INTO teachers (teacher\_id, teacher\_name,teacher\_subject,email) values(5, 'sharma', 'sql', 'sharme@gmail.com');

SELECT \*FROM teachers





CREATE TABLE student (student\_id INT PRIMARY KEY,student\_name VARCHAR(50),age INT,teacher\_id INT,FOREIGN KEY (teacher\_id) REFERENCES teachers(teacher\_id));

drop table teacher;

create table teacher(teachers\_id INT PRIMARY KEY,teachers\_name VARCHAR(50));

INSERT INTO teacher (teachers\_id, teachers\_name)

VALUES (1, 'Mr. Sharma'), (2, 'Ms. Mehta');

INSERT INTO student (student\_id, student\_name, age, teacher\_id)

VALUES

(101, 'Amit', 15, 1),

(102, 'Neha', 16, 2);

select \*from student;

4. Main SQL Commands and Sub-commands (DDL)

1. Define the SQL Data Definition Language (DDL).

SQL Data Definition Language (DDL) refers to a subset of Structured Query Language (SQL) commands used for defining, managing, and modifying the structure and organization of a database. DDL commands are primarily concerned with the database schema and its objects, such as tables, indexes, views, and constraints, rather than the data itself.

Common DDL commands include:

* **CREATE**:

Used to create new database objects, such as databases, tables, views, or indexes.

* **ALTER**:

Used to modify the structure of existing database objects, for example, adding or dropping columns from a table, changing data types, or adding/removing constraints.

* **DROP**:

Used to delete existing database objects, such as tables, views, or indexes, permanently removing them from the database.

2.Explain the CREATE command and its syntax

The CREATE command in SQL is a Data Definition Language (DDL) command used to establish new database objects. Common objects created with this command include databases, tables, views, and indexes.

CREATE DATABASE database\_name;

3. What is the purpose of specifying data types and constraints during table creation?

Specifying data types and constraints during table creation in a database serves several crucial purposes, primarily focused on maintaining data integrity and optimizing database performance:

* **Data Types:**

Data types define the kind of data a column can store (e.g., integers, text, dates, decimals). This ensures that only valid data is entered, preventing errors like trying to store text in a numeric column.

* **Constraints:**

Constraints enforce rules on the data, ensuring its accuracy and reliability. Examples include:

* + **NOT NULL:** Ensures a column cannot contain empty values.
  + **UNIQUE:** Guarantees that all values in a column are distinct.
  + **PRIMARY KEY:** Uniquely identifies each row in a table, ensuring no duplicates and providing a primary reference point.
  + **FOREIGN KEY:** Establishes relationships between tables, enforcing referential integrity by ensuring that values in a foreign key column match existing values in the referenced primary key column.

5. ALTER Command

1. What is the use of the ALTER command in SQL?

The ALTER command in SQL is a Data Definition Language (DDL) command used to modify the structure of an existing database object, primarily tables. It allows for various structural changes without recreating the entire object.

Here are the primary uses of the ALTER command, specifically in the context of ALTER TABLE:

* **Adding Columns:** New columns can be added to an existing table, specifying their name, data type, and any constraints.
* **Dropping Columns:** Existing columns can be removed from a table.
* **Modifying Column Definitions:** The data type, size, or other properties of an existing column can be changed. For example, changing a VARCHAR column's length or altering a column's NULLability.
* **Adding/Dropping Constraints:** Constraints like PRIMARY KEY, FOREIGN KEY, UNIQUE, NOT NULL, and CHECK can be added to or removed from columns or the table itself.
* **Renaming Tables or Columns:** The names of tables or individual columns within a table can be changed.

2. How can you add, modify, and drop columns from a table using ALTER?

1. Adding Columns:

To add a new column to an existing table, the ADD COLUMN clause is used.

ALTER TABLE Employees

ADD COLUMN HireDate DATE;

2. Modifying Columns:

To modify an existing column's data type, size, or constraints, the MODIFY COLUMN (or ALTER COLUMN in some systems) clause is used.

ALTER TABLE Products

MODIFY COLUMN Price DECIMAL(10, 2);

3. Dropping Columns:

To remove an existing column from a table, the DROP COLUMN clause is used.

ALTER TABLE Customers

DROP COLUMN OldAddress;

6. DROP Command

1. What is the function of the DROP command in SQL?

The DROP command in SQL is a Data Definition Language (DDL) command used to remove entire database objects from a relational database management system (RDBMS). This command permanently deletes the specified object and all its associated data and structures.

DROP DATABASE database\_name;

2. What are the implications of dropping a table from a database?

Dropping a table from a database has significant consequences, primarily resulting in the permanent deletion of both the table's data and its structure, along with associated indexes, constraints, triggers, and privileges. This action is irreversible without a backup, and it can impact database performance and functionality.

7. Data Manipulation Language (DML)

1. Define the INSERT, UPDATE, and DELETE commands in SQL.

The INSERT, UPDATE, and DELETE commands in SQL are fundamental Data Manipulation Language (DML) statements used to manage data within a database.

* **INSERT:** The INSERT command is used to add new rows (records) into an existing table. It allows for the specification of values for all columns in the new row, or for a subset of columns, with the remaining columns taking their default values or NULL if allowed.

INSERT INTO table\_name (column1, column2, ...)

VALUES (value1, value2, ...);

* **UPDATE:** The UPDATE command is used to modify existing data within one or more rows of a table. It allows for changes to specific columns and can be targeted to particular rows using a WHERE clause.

UPDATE table\_name

SET column1 = new\_value1, column2 = new\_value2, ...

WHERE condition;

* **DELETE:** The DELETE command is used to remove one or more rows from a table. It can be used to remove all rows from a table or a specific subset of rows based on a WHERE clause.

DELETE FROM table\_name

WHERE condition;

2. What is the importance of the WHERE clause in UPDATE and DELETE operations?

The WHERE clause in UPDATE and DELETE operations is crucial for specifying which rows in a table should be affected by the operation. Its importance lies in preventing unintended and potentially catastrophic data modifications or deletions.

8. Data Query Language (DQL)

1. What is the SELECT statement, and how is it used to query data?

The SELECT statement is a fundamental command in Structured Query Language (SQL) used to retrieve data from one or more tables within a database. It allows users to specify which columns to display and apply conditions to filter the results, returning the data in a result set, also known as a result table.

How it is used to query data:

**Specifying Columns:**

The SELECT clause determines which columns from a table will be included in the result set.

To select all olumns: SELECT \* FROM table\_name;

To select specific columns: SELECT column1, column2 FROM table\_name;

**Specifying Tables:**

The FROM clause indicates the table(s) from which the data is to be retrieved.

Example: SELECT column1 FROM table\_name;

**Filtering Data (Optional):**

The WHERE clause is used to apply conditions, filtering the rows returned based on specified criteria.

Example: SELECT \* FROM table\_name WHERE column\_name = 'value';

The ORDER BY clause sorts the result set based on one or more columns in ascending (ASC) or descending (DESC) order.

Example: SELECT \* FROM table\_name ORDER BY column\_name DESC;

**Grouping Data (Optional):**

The GROUP BY clause groups rows that have the same values in specified columns into summary rows, often used with aggregate functions (e.g., COUNT, SUM, AVG).

Example: SELECT column1, COUNT(\*) FROM table\_name GROUP BY column1;

**Limiting Results (Optional):**

The LIMIT clause (in some SQL dialects like MySQL) restricts the number of rows returned.

Example: SELECT \* FROM table\_name LIMIT 10;

The SELECT statement, in its various forms, provides a powerful and flexible way to extract specific information from a database according to user requirements.

2. Explain the use of the ORDER BY and WHERE clauses in SQL queries

In SQL queries, the WHERE and ORDER BY clauses serve distinct but often complementary purposes:

1. WHERE Clause:

The WHERE clause is used to filter records based on a specified condition. It extracts only those rows from a table that satisfy the given criteria. This clause is fundamental for retrieving a specific subset of data from a larger dataset.

* **Purpose:** To filter rows based on a condition, returning only those that meet the criteria.
* **Placement:** Typically placed after the FROM clause and before GROUP BY or ORDER BY.

SELECT CustomerName, City

FROM Customers

WHERE Country = 'Mexico';

2. ORDER BY Clause:

The ORDER BY clause is used to sort the result set of a query based on one or more columns. It arranges the rows in either ascending (default) or descending order, providing a structured and organized presentation of the data.

* **Purpose:** To sort the retrieved rows in a specified order (ascending or descending).
* **Placement:** Typically placed after WHERE, GROUP BY, or HAVING clauses, as it operates on the final result set.
* **Example:**

SELECT CustomerName, City, Country

FROM Customers

ORDER BY Country ASC, City DESC;

* WHERE: selects which rows to include in the result.
* ORDER BY: determines how the included rows are arranged.

9. Data Control Language (DCL)

1. What is the purpose of GRANT and REVOKE in SQL?

In SQL, GRANT and REVOKE are Data Control Language (DCL) commands used to manage dataase security and access control by defining and removing user privileges.

The GRANT command is used to assign specific permissions or privileges to users or roles within a database. These privileges determine what actions a user or role can perform on database objects (like tables, views, or stored procedures) or system-level operations.

GRANT SELECT, INSERT ON Employees TO AnalystUser;

The REVOKE command is used to remove previously granted permissions or privileges from users or roles. This allows database administrators to restrict or remove access to specific database objects or functionalities.

REVOKE INSERT ON Employees FROM AnalystUser;

* GRANT provides authorization and access to database resources.
* REVOKE removes or restricts that authorization and access.

2. How do you manage privileges using these commands?

Privileges are managed using commands that grant or remove specific permissions for users or roles within a system, particularly in database management systems like SQL. The primary commands for this purpose are GRANT and REVOKE.

The GRANT command assigns specific privileges to a user or role, allowing them to perform certain actions or access particular objects. syntax.

GRANT privilege\_type ON object\_name TO user\_name\_or\_role\_name;

10. Transaction Control Language (TCL)

1. What is the purpose of the COMMIT and ROLLBACK commands in SQL?

The SQL COMMIT and ROLLBACK commands are integral to managing transactions and ensuring data integrity within a database. They are used to control the state of changes made during a transaction.

* **COMMIT:**

The COMMIT command is used to permanently save all changes made during the current transaction to the database. Once a transaction is committed, the changes become durable and visible to other users and subsequent transactions. This command effectively marks the successful completion of a transaction, making all modifications irreversible (without explicitly performing new operations to undo them).

* **ROLLBACK:**

The ROLLBACK command is used to undo all changes made during the current transaction since the last COMMIT or BEGIN TRANSACTION statement. If an error occurs during a transaction, or if a user decides to abandon the changes, ROLLBACK restores the database to its state before the transaction began. This ensures that incomplete or erroneous operations do not corrupt the database and maintains data consistency.

2. Explain how transactions are managed in SQL databases.

Transactions in SQL databases are managed to ensure data integrity and consistency, especially when multiple operations need to be treated as a single, atomic unit of work. This management adheres to the ACID properties: Atomicity, Consistency, Isolation, and Durability.

* BEGIN TRANSACTION (or START TRANSACTION):

This command marks the explicit beginning of a transaction. All subsequent DML (Data Manipulation Language) statements (e.g., INSERT, UPDATE, DELETE) become part of this transaction.

* COMMIT TRANSACTION (or COMMIT):

This command finalizes a transaction, making all changes made within it permanent in the database. Once committed, the changes cannot be undone by a ROLLBACK.

* ROLLBACK TRANSACTION (or ROLLBACK):

This command undoes all changes made during the current transaction, effectively reverting the database to its state before the transaction started. This is typically used when an error occurs or a condition is not met.

* SAVEPOINT:

This command allows for partial rollbacks within a transaction. You can set a savepoint and then roll back to that specific point, rather than rolling back the entire transaction.

BEGIN TRANSACTION;

-- Perform operations within the transaction

UPDATE Accounts SET Balance = Balance - 100 WHERE AccountID = 123;

UPDATE Accounts SET Balance = Balance + 100 WHERE AccountID = 456;

-- Check for conditions or potential errors

-- IF error\_condition THEN

-- ROLLBACK; -- Undo all changes

-- ELSE

-- COMMIT; -- Make changes permanent

-- END IF;

11. SQL Joins

1. Explain the concept of JOIN in SQL. What is the difference between INNER JOIN, LEFT JOIN, RIGHT JOIN, and FULL OUTER JOIN?

In SQL, a JOIN clause combines rows from two or more tables based on a related column between them. This allows you to retrieve data that is logically connected across different tables in your database.

Here's a breakdown of the different JOIN types:

* INNER JOIN:
  + Returns only the rows that have matching values in both tables.
  + It effectively finds the intersection of the two tables based on the join condition.
  + Rows that do not have a match in the other table are excluded from the result set.

SELECT columns

FROM TableA

INNER JOIN TableB ON TableA.common\_column = TableB.common\_column;

* LEFT JOIN (or LEFT OUTER JOIN):
  + Returns all rows from the left table (the first table specified in the FROM clause).
  + It also returns the matching rows from the right table.
  + If there are no matching rows in the right table for a given row in the left table, the columns from the right table will contain NULL values.

SELECT columns

FROM TableA

LEFT JOIN TableB ON TableA.common\_column = TableB.common\_column;

* RIGHT JOIN (or RIGHT OUTER JOIN):
  + Returns all rows from the right table (the second table specified in the FROM clause).
  + It also returns the matching rows from the left table.
  + If there are no matching rows in the left table for a given row in the right table, the columns from the left table will contain NULL values.

SELECT columns

FROM TableA

RIGHT JOIN TableB ON TableA.common\_column = TableB.common\_column;

FULL OUTER JOIN (or FULL JOIN):

* + Returns all rows when there is a match in either the left or the right table.
  + It combines the results of both a LEFT JOIN and a RIGHT JOIN.
  + If a row in one table does not have a match in the other table, the columns from the non-matching table will contain NULL values.

SELECT columns

FROM TableA

FULL OUTER JOIN TableB ON TableA.common\_column = TableB.common\_column;