T-SQL
T-SQL Statement Types

- DML(Data Manipulation Language) – Used to retrieve, store, modify, delete, insert and update data in database.

Eg - SELECT, UPDATE, INSERT

- DDL(Data definition Language) - Used to create and modify the **structure of database objects** in database.

Eg - CREATE, ALTER, DROP

- DCL (Data Control Language) – These SQL commands are used for managing security of database objects

Eg - GRANT, REVOKE

DATA TYPES

- SQL Server data type is an attribute that specifies types of data of any object.
- Each column, variable and expression has related data type in SQL Server.
- These data types can be used while creating tables. You can choose a particular data type for a table column based on your requirement

Numeric (int, decimal, money)	Unicode character strings(nchar, nvarchar)
	Binary strings(Yes/No, 0/1, Active/Inactive)
Date and time (date , datetime YY-MM-DD hh:mm:ss)	Other data types(xml,cursor)
Character strings (Char, Varchar)	

- nchar and nvarchar can store Unicode characters.
- char and varchar cannot store Unicode characters.
- char and nchar are fixed-length which will reserve storage space for number of characters you specify even if you don't use up all that space.
- varchar and nvarchar are variable-length which will only use up spaces for the characters you store. It will not reserve storage like char or nchar.

SELECT STATEMENT

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

```
SELECT column1, column2, columnN FROM table_name;

SELECT * FROM table_name;
```

- Example

```
SELECT ID, NAME, SALARY FROM CUSTOMERS;
```

CREATING AND DROP TABLE

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

```
CREATE TABLE table_name(
    column1 datatype,
    column2 datatype,
    column3 datatype,
    .....
    columnN datatype,
    PRIMARY KEY( one or more columns ));
```

- Example

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

- The SQL Server **DROP TABLE** statement is used to delete a table.
- Syntax

```
DROP TABLE table_name;
```

INSERT INTO TABLE

- The SQL Server **INSERT INTO** statement is used to add new rows of data to a table in the database.
- Syntax

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

- Example

```
INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (1, ‘John', 32, ‘Silver Spring’, 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (2, ‘Sara', 25, ‘DC', 1500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (3, ‘Ruth', 23, ‘Omaha', 2000.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (4, ‘Smith’, 25, ‘New York', 6500.00 );

INSERT INTO CUSTOMERS (ID,NAME,AGE,ADDRESS,SALARY)
VALUES (5, ‘Ava', 27, ‘Boston', 8500.00 );
```

WHERE CLAUSE

- The MS SQL Server **WHERE** clause is used to specify a condition while fetching the data from single table or joining with multiple tables...
- If the given condition is satisfied, only then it returns a specific value from the table. You will have to use WHERE clause to filter the records and fetch only necessary records.
- The WHERE clause is not only used in SELECT statement, but it is also used in UPDATE, DELETE statement, etc., which we would examine in subsequent chapters.
- Syntax

```
SELECT column1, column2, columnN
FROM table_name
WHERE [condition]
```

- Example

```
SELECT ID, NAME, SALARY
FROM CUSTOMERS
WHERE SALARY > 2000;
```

UPDATE CLAUSE

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

```
UPDATE table_name
SET column1 = value1, column2 = value2...., columnN = valueN
WHERE [condition];
```

- Example

```
UPDATE CUSTOMERS
SET ADDRESS = ‘Portland'
WHERE ID = 6;
```

DELETE CLAUSE

- The SQL Server **DELETE** Query is used to delete the existing records from a table.
- You have to use WHERE clause with DELETE query to delete selected rows, otherwise all the records would be deleted.
- Syntax

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

- Example

```
DELETE FROM CUSTOMERS
WHERE ID = 3;
```

LIKE CLAUSE

- The MS SQL Server **LIKE** clause is used to compare a value to similar values using wildcard operators.
  The percent sign (%)
- The percent sign represents zero, one, or multiple characters.
- Syntax

```
SELECT  column-list FROM table_name
WHERE column LIKE 'XXXX%'
```

Example

```
SELECT *FROM CUSTOMERS
WHERE SALARY  LIKE '200%'
```

ORDER BY CLAUSE

- The MS SQL Server **ORDER** BY clause is used to sort the data in ascending or descending order, based on one or more columns. Some database sort query results in ascending order by default.
- Syntax

```
SELECT column-list
FROM table_name
[WHERE condition]
[ORDER BY column1, column2, .. columnN] [ASC | DESC];
```

- Example

```
SELECT * FROM CUSTOMERS
ORDER BY NAME, SALARY ASC

SELECT * FROM CUSTOMERS
ORDER BY NAME DESC
```

Group Functions:

- Group functions are built-in SQL functions that operate on groups of rows and return one value for the entire group.
- **These functions are:**
- COUNT
- MAX
- MIN
- AVG
- SUM
- DISTINCT

COUNT ()

- **COUNT ():** This function returns the number of rows in the table that satisfies the condition specified in the WHERE condition.
- If the WHERE condition is not specified, then the query returns the total number of rows in the table.
- **For Example:** If you want the number of students in a particular subject, the query would be:

```
SELECT COUNT (*) as 'Number of Student in History'
From student_detail
WHERE subject = 'History';
```

MAX() AND MIN()

- **MAX():** This function is used to get the maximum value from a column.
- Example, to get the maximum salary drawn by an employee, the query would be:

```
SELECT MAX (salary) as 'Highest Salary'
FROM employee;
```

- **MIN():** This function is used to get the minimum value from a column.
- Example, to get the minimum salary drawn by an employee, the query would be:

```
SELECT MIN (salary) as 'Lowest Salary'
FROM employee;
```

AVG() AND SUM()

- AVG(): This function is used to get the average value of a numeric column.
- Example, to get the average salary drawn by an employee, the query would be:

```
SELECT AVG (salary) as 'Average Salary'
 FROM employee
```

- SUM(): This function is used to get the sum of a numeric column
- Example, to get the total salary drawn by all employees, the query would be:

```
SELECT SUM (salary) as 'Total Salary'
FROM employee
```

DISTINCT CLAUSE

- The MS SQL Server **DISTINCT** keyword is used in conjunction with SELECT statement to eliminate all the duplicate records and fetching only unique records.
- There may be a situation when you have multiple duplicate records in a table. While fetching such records, it makes more sense to fetch only unique records instead of fetching duplicate records..
- Syntax

```
SELECT DISTINCT column1, column2,.....columnN
FROM table_name
WHERE [condition]
```

- Example

```
SELECT DISTINCT SALARY FROM CUSTOMERS
ORDER BY SALARY
```

Joins CLAUSE

- The MS SQL Server **Joins** clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

  Example

```
SELECT ID, NAME, AGE, AMOUNT
FROM CUSTOMERS
inner join  ORDERS
ON CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

MS SQL Server Join Types –

There are different types of joins available in MS SQL Server –

INNER JOIN – Returns records that have matching values in both tables

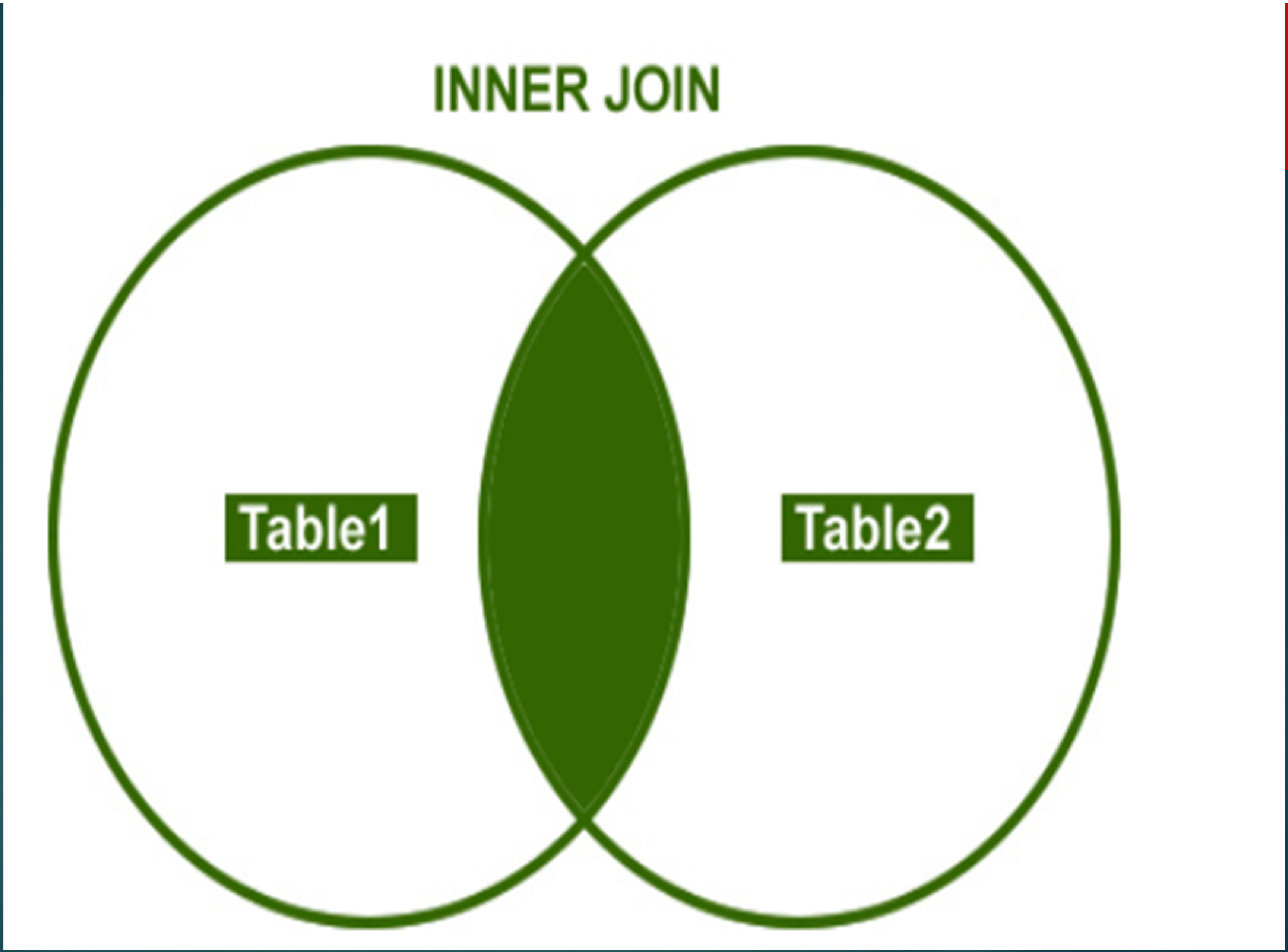
LEFT JOIN – Returns all records from the left table, and the matched records from the right table

RIGHT JOIN – Returns all records from the right table, and the matched records from the left table

.**FULL JOIN** – returns all the rows from the joined tables, whether they are matched or not

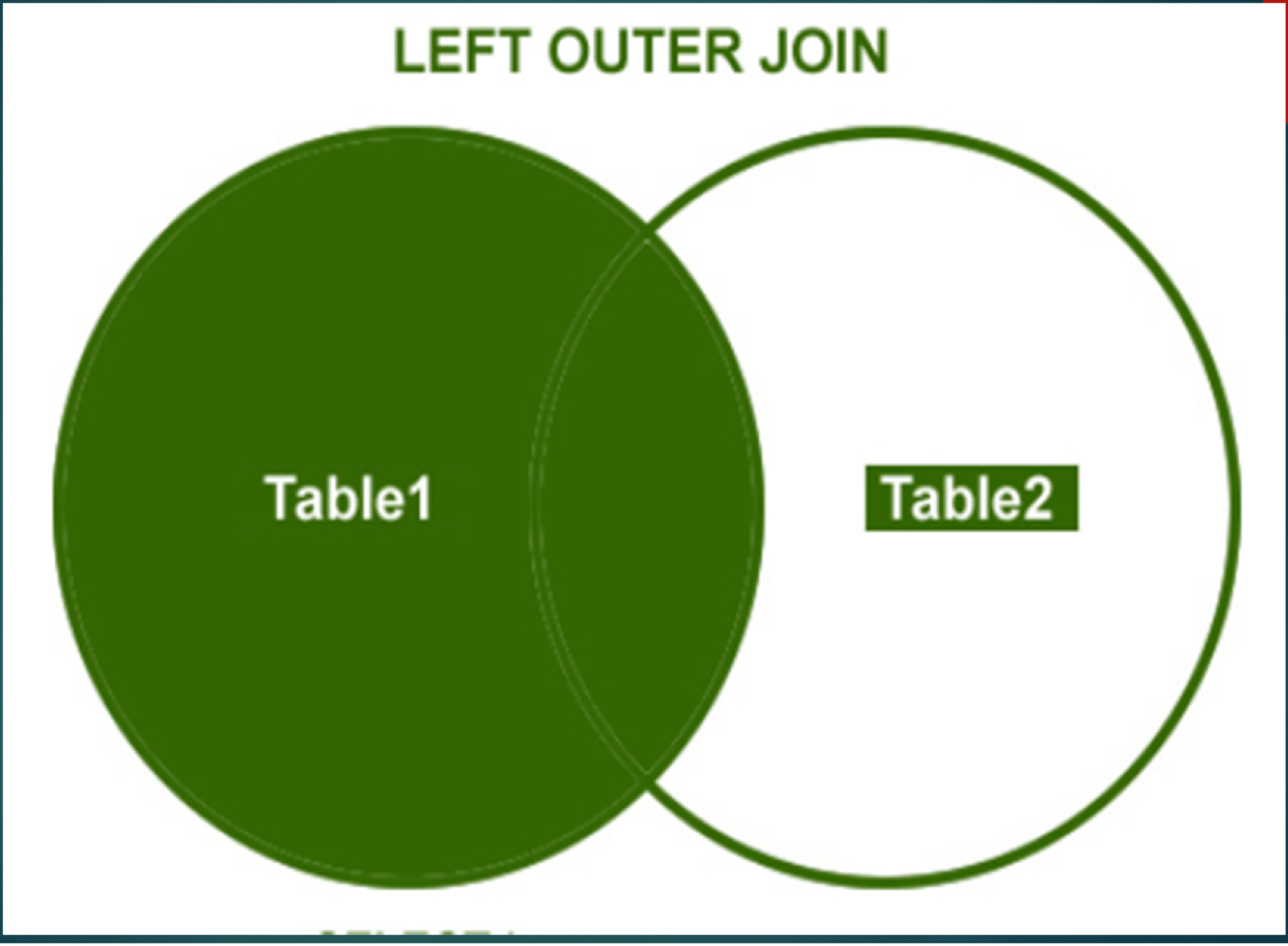
INNER JOIN:

- This join returns rows when there is at least one match in both the tables.
- SQL Server INNER JOINS return all rows from multiple tables where the join condition is met.
- SELECT *column_name(s)*
  FROM *table1*
  INNER JOIN *table2*
  ON *table1.column_name* = *table2.column_name;*


INNER JOIN
Table1
Table2

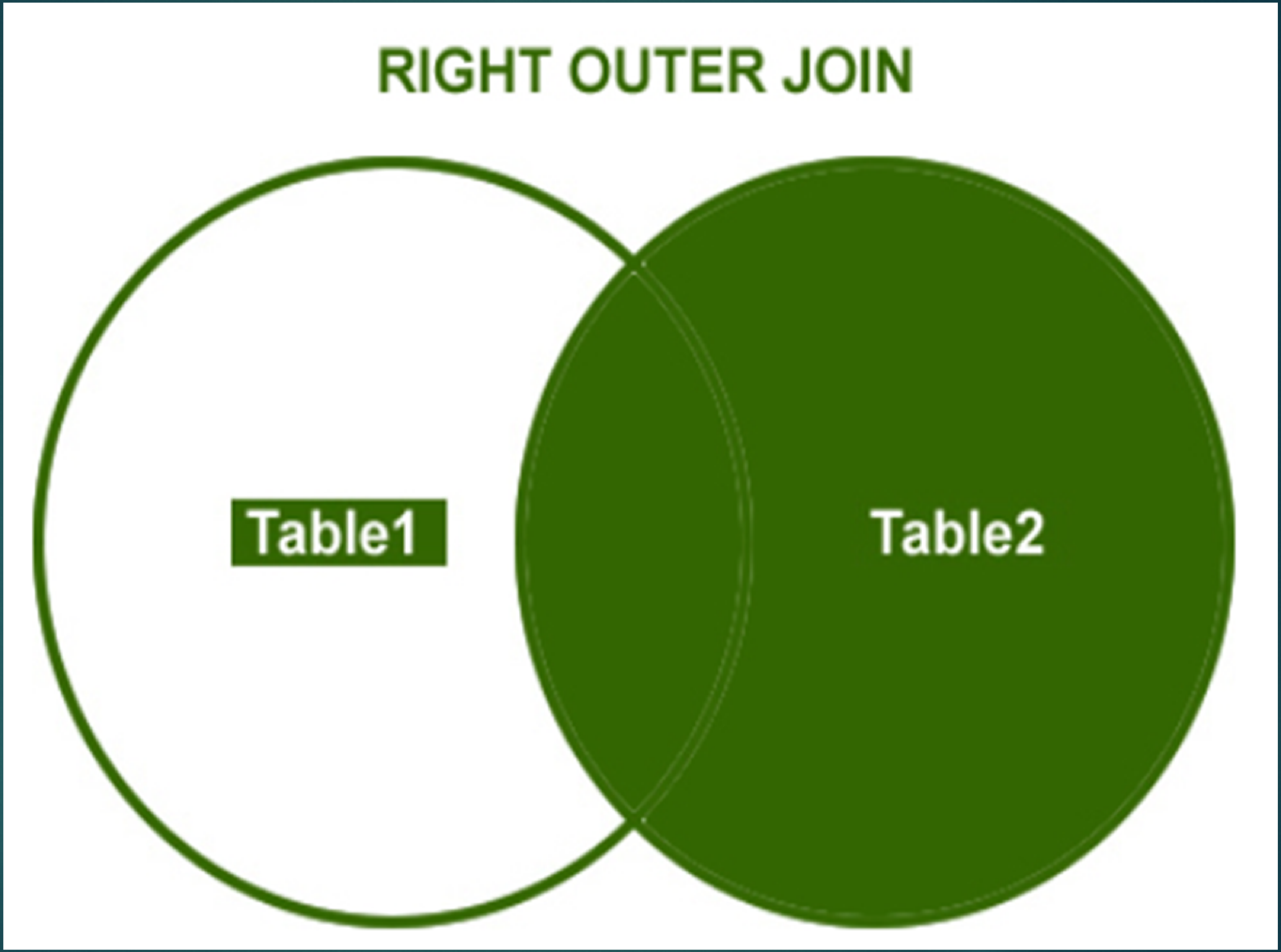
LEFT OUTER JOIN:

- This type of join returns all rows from the LEFT-hand table specified in the ON condition and **only** those rows from the other table where the joined fields are equal (join condition is met).
- This join returns all the rows from the left table in conjunction with the matching rows from the right table.
- If there are no columns matching in the right table, it returns NULL values.
- SELECT *column_name(s)*
  FROM *table1*
  LEFT JOIN *table2*
  ON *table1.column_name* = *table2.column_name;*


LEFT OUTER JOIN
Table1
Table2

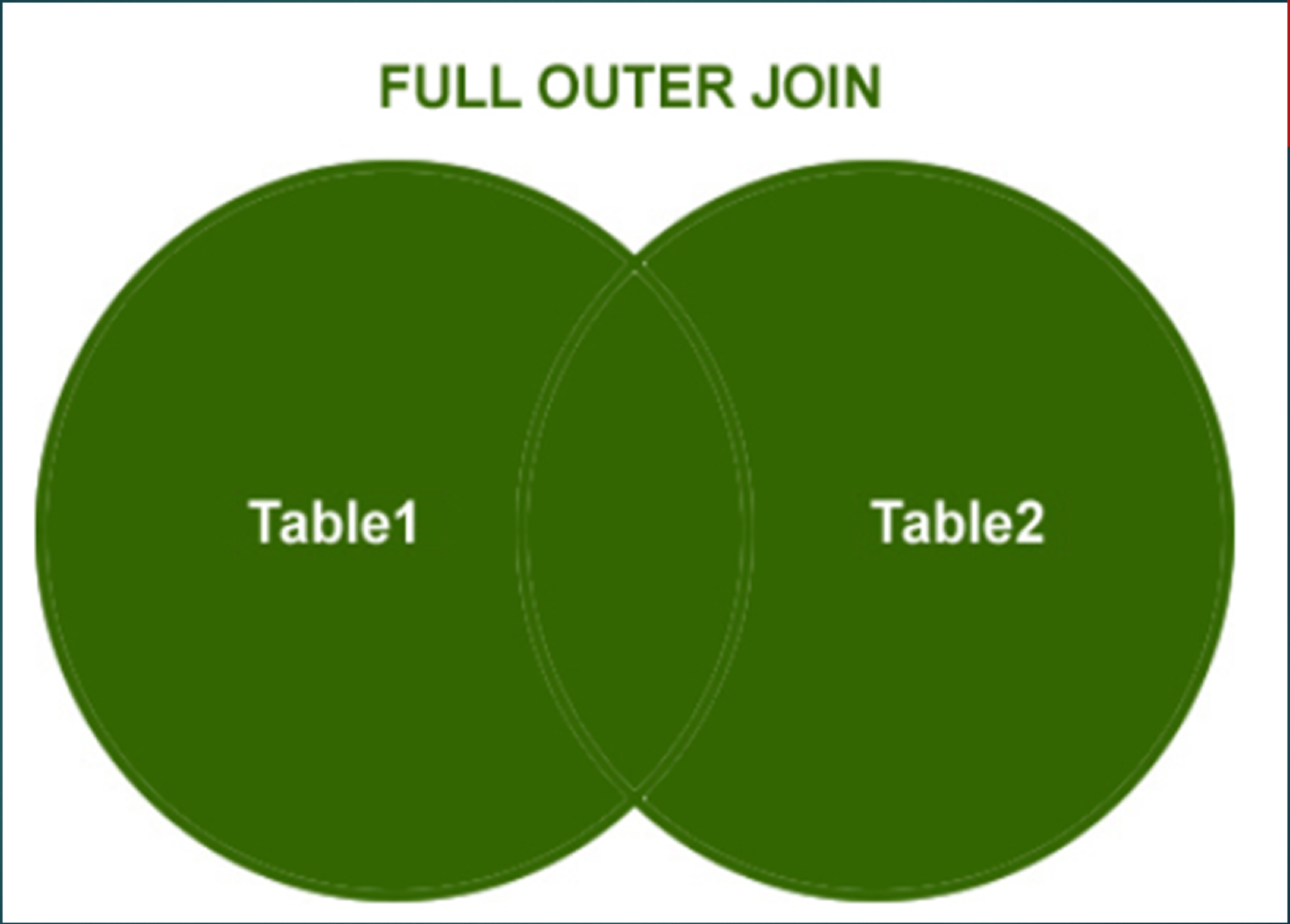
RIGHT OUTER JOIN:

- This join returns all the rows from the right table in conjunction with the matching rows from the left table.
- If there are no columns matching in the left table, it returns NULL values.
- This type of join returns all rows from the RIGHT-hand table specified in the ON condition and **only** those rows from the other table where the joined fields are equal (join condition is met).
- SELECT *column_name(s)*
  FROM *table1*
  RIGHT JOIN *table2*
  ON *table1.column_name* = *table2.column_name;*


RIGHT OUTER JOIN
Table1
Table2

FULL OUTER JOIN:

- This join combines left outer join and right outer join.
- It returns row from either table when the conditions are met and returns null value when there is no match.
- This type of join returns all rows from the LEFT-hand table and RIGHT-hand table with nulls in place where the join condition is not met.
- SELECT *column_name(s)*
  FROM *table1*
  FULL OUTER JOIN *table2*
  ON *table1.column_name = table2.column_name*
  WHERE *condition;*


FULL OUTER JOIN
Table1
Table2