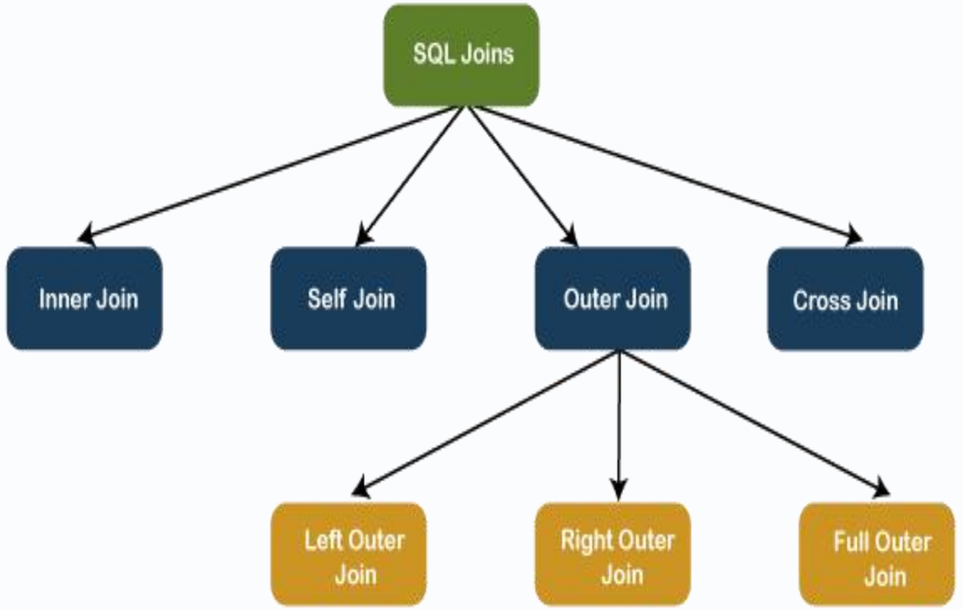
SQL JOINs

SQL JOINs are a powerful feature in the Structured Query Language (SQL) that allow you to combine data from multiple tables based on a related column between them. JOINs are essential for building complex queries and unlocking the full potential of relational databases.



Implementing SQL JOINs

1

Identify Relationships

Understand the data model and the relationships between your tables. This will help you determine which type of JOIN to use.

2

Write the JOIN Clause

Construct the appropriate JOIN clause (INNER, LEFT, RIGHT, FULL OUTER, or CROSS) and specify the ON condition to link the tables.

3

Select Columns

Decide which columns you want to retrieve from the joined tables, and include them in the SELECT statement.

4

Test and Refine

Execute the query and analyze the results. Adjust the JOIN conditions or column selections as needed to get the desired output.

Syntax of Joins

ANSI SQL Syntax

The ANSI SQL standard syntax for a join is:

```
SELECT *
FROM table1
JOIN table2;
```

Legacy SQL Syntax

Some older SQL dialects may use the following syntax for a join:

```
SELECT *
FROM table1, table2;
```

What are SQL JOINs?

Combining Data

SQL JOINs enable you to combine data from two or more tables based on a common column or field. This allows you to retrieve a more complete set of information by merging data that is stored across multiple tables in your database.

Relationship Types

JOINs are used to establish relationships between tables, such as one-to-one, one-to-many, or many-to-many. These relationships define how the data in the tables is connected and how it should be combined.

Efficient Querying

By using JOINs, you can write more efficient and powerful SQL queries that retrieve the exact data you need, without having to manually combine the results from multiple separate queries.

Types of Joins

1 Inner Join

Returns only the rows that have matching values in both tables.

2 Left Join

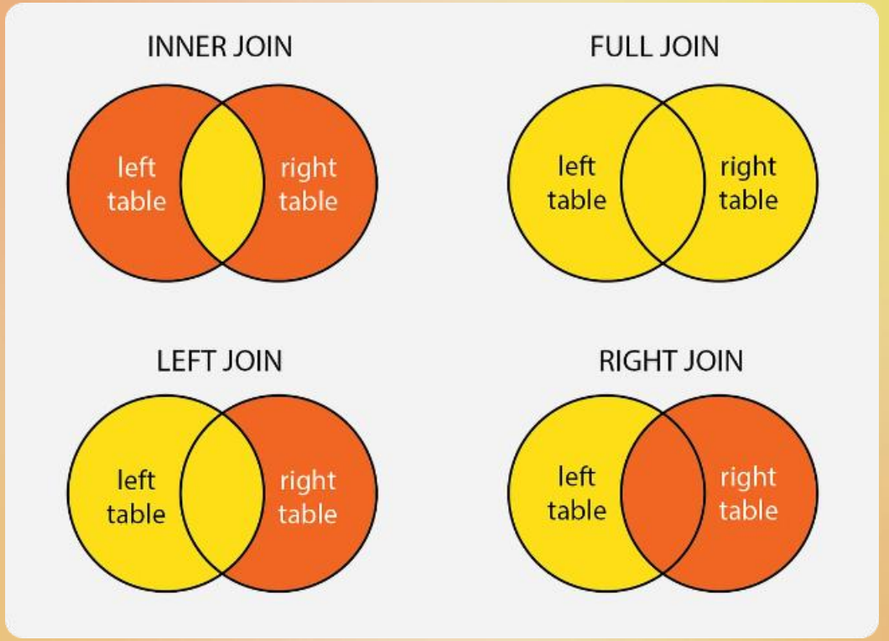
Returns all rows from the left table, and the matching rows from the right table.

3 Right Join

Returns all rows from the right table, and the matching rows from the left table.

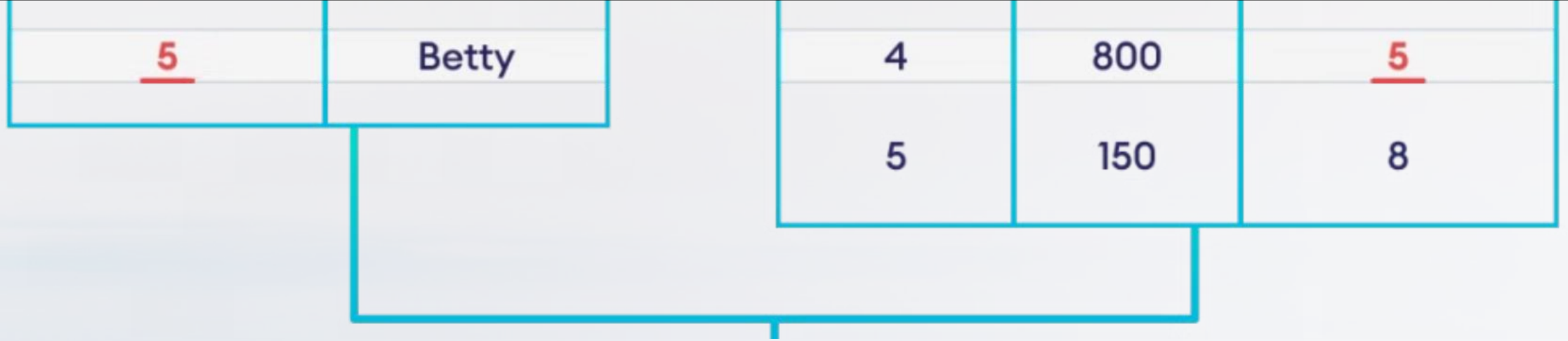
4 Outer Join

Returns all rows from both tables, whether or not there is a match.



Introduction to Cross Join

A CROSS JOIN, also known as a Cartesian Join, combines all rows from two tables without any condition. The result set includes all possible combinations of rows from the two tables. The number of rows in the result set is the product of the number of rows in each table.



Syntax for Cross Join

The syntax for a cross join is:

```
SELECT columns
FROM table1
CROSS JOIN table2;
```

This means that each row from the first table is combined with each row from the second table

Cross Join

Example

```
SELECT  *
 FROM              A
CROSS JOIN      B ;
```

A

n
1
2
3

B

c
x
y
z

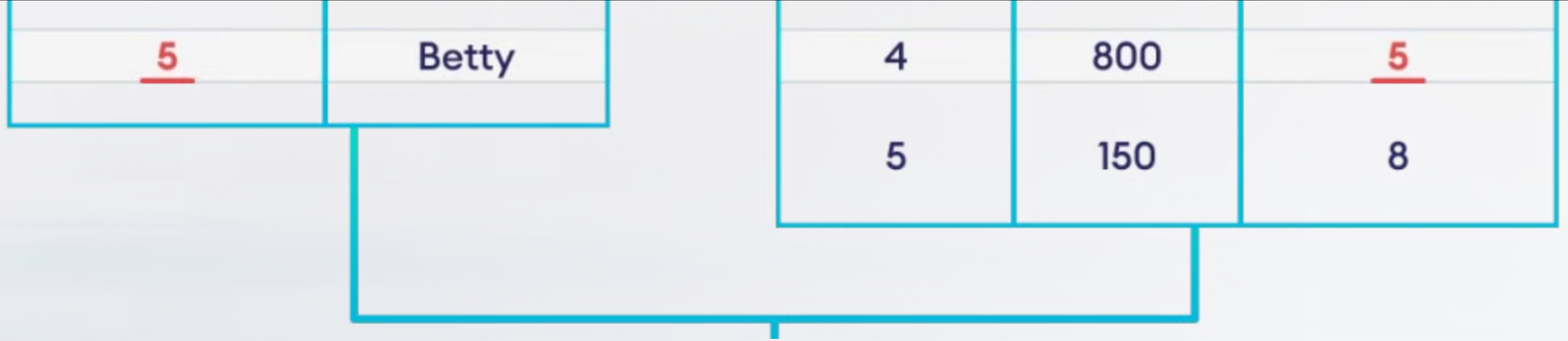
```
SELECT *
FROM A
CROSS JOIN B
```

A x B

n	c
1	x
1	y
1	z
2	x
2	y
2	z
3	x
3	y
3	z

Introduction to Inner Join

Inner join is a type of SQL join that returns only the rows that have matching values in both of the joined tables. In other words, it combines the data from two tables based on a common column, and the result includes only the records that have a match in both tables. This is the most basic and widely used join type, as it allows you to efficiently retrieve relevant data by linking information from multiple sources.



Syntax for Inner Join

The syntax for a inner join is:

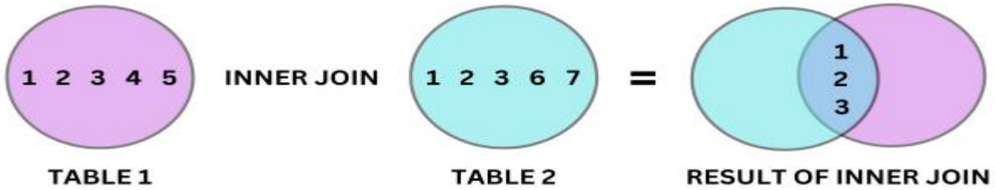
```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.common_column = table2.common_column;
```

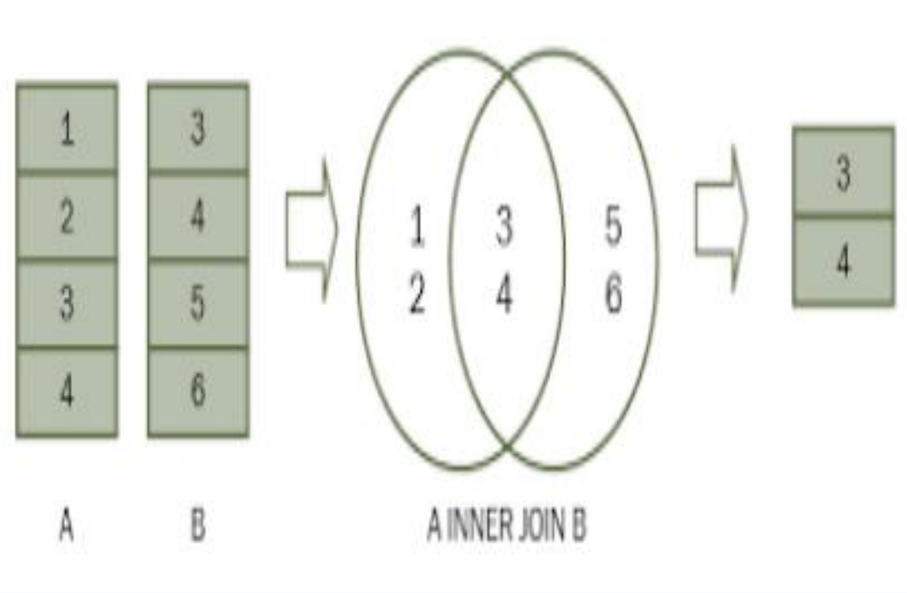
This will return only the rows that have matching values in both(table1 and table2) of the joined tables..

Inner Join

Example

```
SELECT
        A
FROM
        A
INNER JOIN
        B
ON A.id = B.ID;
```





Q1) List all employees along with their family member names.

Q2) Display empno, deptno,deptname for all employees

Q3) Find employees who have a passport.

Q4) Find employees working in the 'Engineering' department.

Q5) Find employees working in the 'Finance' department.

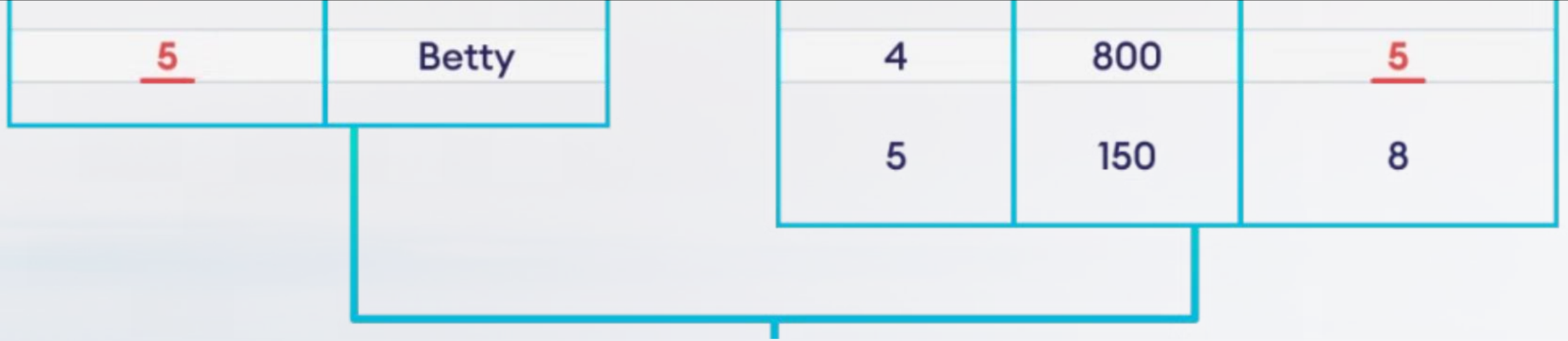
Q6) List all departments with the Average Salaray of employees in each.

Q7) List all departments with the count of employees in each.

Q8) List employees along with their passport numbers and PAN card numbers

Introduction to Left Join

A left join, also known as a left outer join, left join is a type of SQL join that returns all rows from the left table, along with the matching rows from the right table. It ensures that all records from the left table are included in the result set, even if there are no matches in the right table.



Syntax for Left Join

The syntax for a left join is:

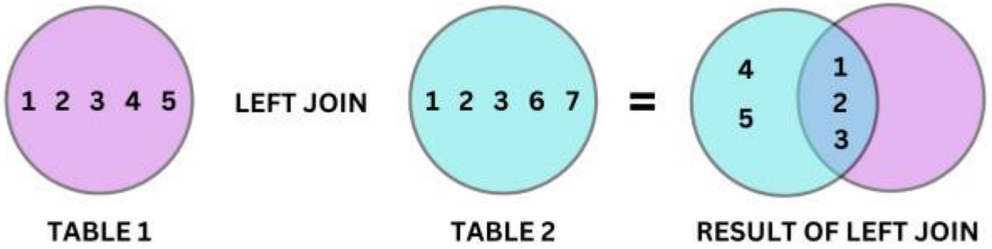
```
SELECT column1, column2, ...
FROM table1
LEFT JOIN table2
ON table1.column = table2.column;
```

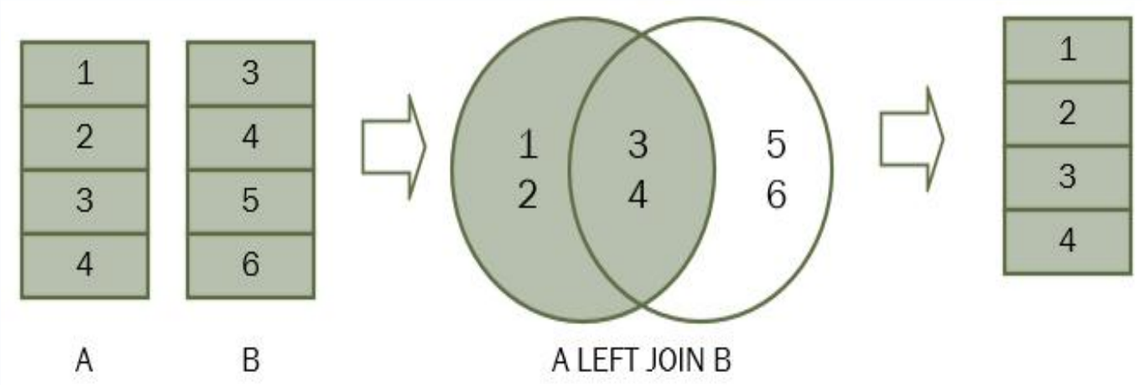
This will return all rows from the left table (table1), along with the matching rows from the right table (table2).

Left Join

Example

```
SELECT
        A
FROM
        A
LEFT JOIN
        B
ON B.n = A.n;
```





Q) Retrieve all employees and their family members.

Q) Retrieve all employees and their passport members.

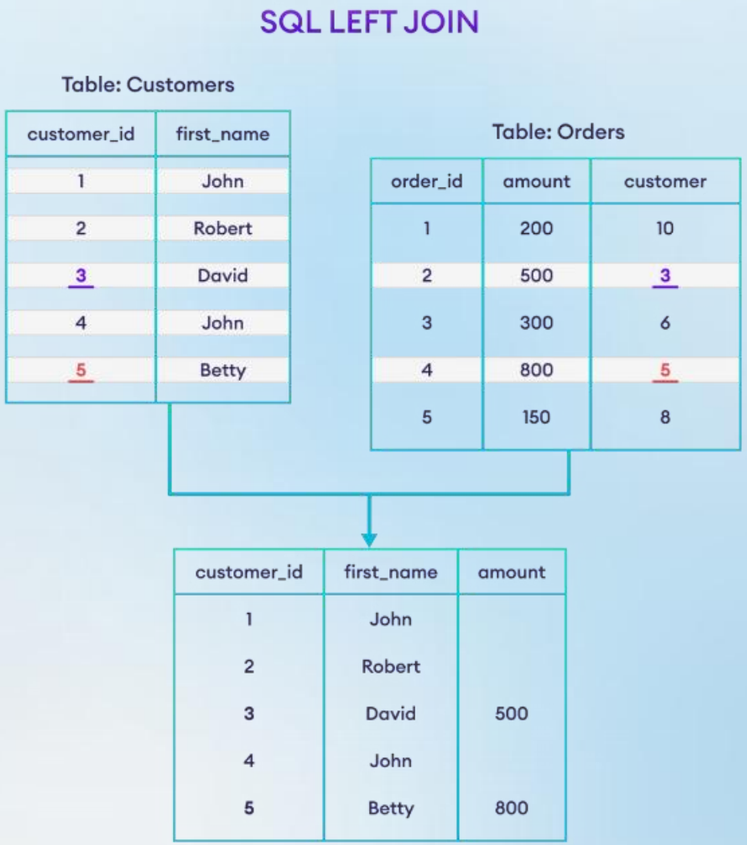
Q) To display empno, deptno,dname for all employees with sal>2000

Q) List all employees along with Pancard

Q) List all departments with the count of employees in each.

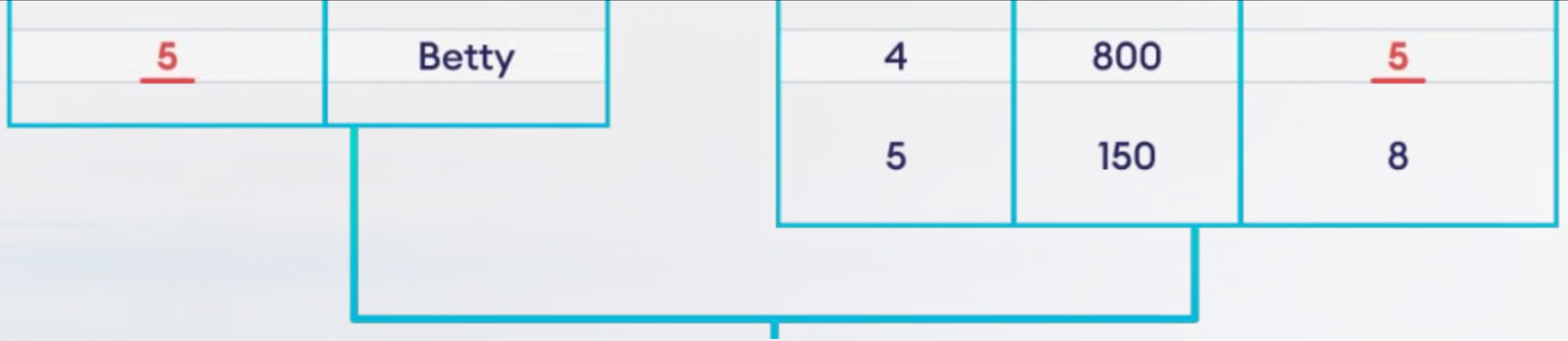
Q) Count of all employees who have passport

Q) List all employees dont have Pancard.



Introduction to Right Join

A right join, also known as a right outer join, right join is a type of SQL join that returns all rows from the right table, along with the matching rows from the left table. It ensures that all records from the right table are included in the result set, even if there are no matches in the right table.



Syntax for Right Join

The syntax for a right join is:

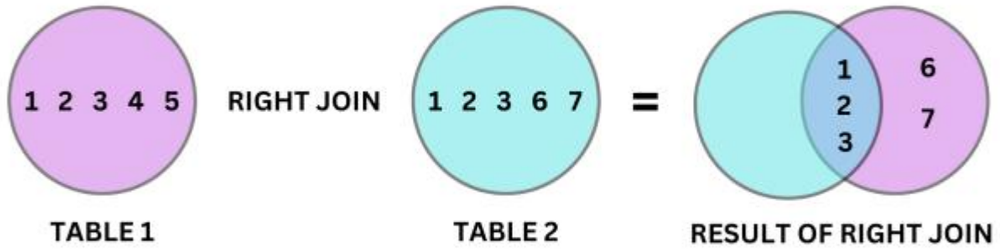
```
SELECT column1, column2, ...
FROM table1
RIGHT JOIN table2
ON table1.column = table2.column;
```

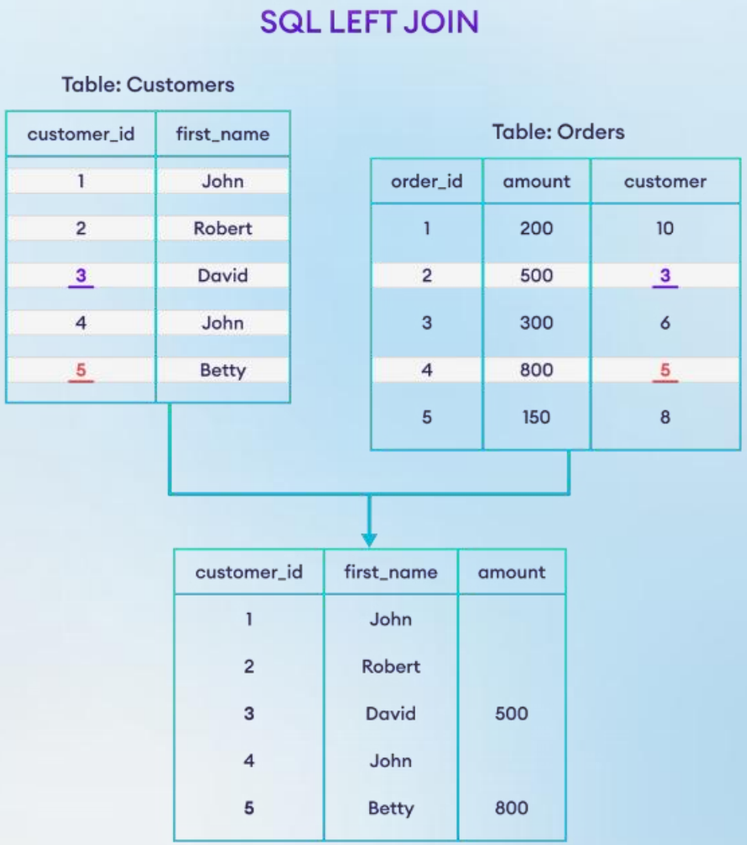
This will return all rows from the right table (table2), along with the matching rows from the left table (table1).

Right Join

Example

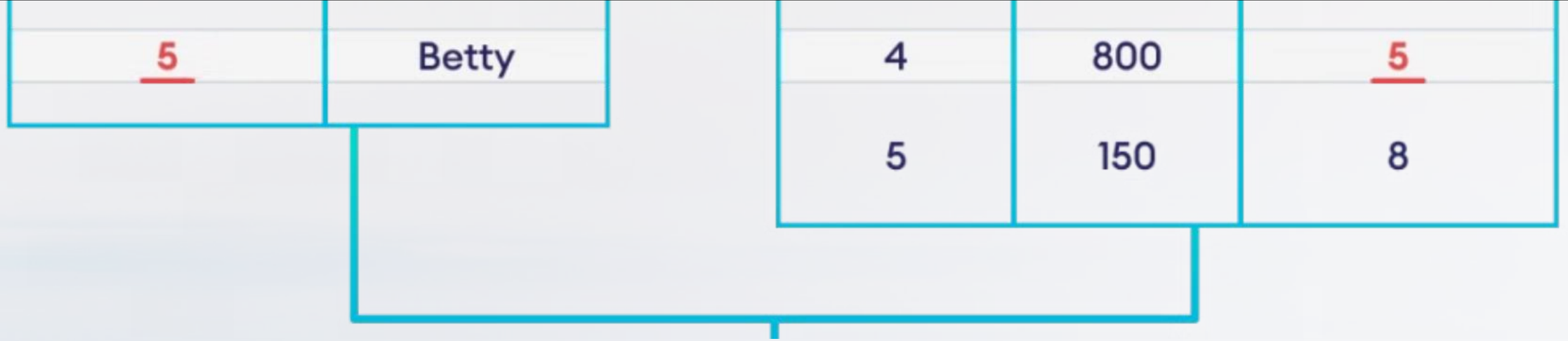
```
SELECT
        A
FROM
        A
RIGHT JOIN
        B
ON B.n = A.n;
```





Introduction to Outer Join

In SQL, an outer join is a type of join operation that allows you to combine rows from two or more tables based on a related column between them. It includes unmatched rows from one or both tables, filling in NULL values for columns where no match is found. Outer joins are categorized into three types: left outer join, right outer join, and full outer join



Syntax for Outer Join

The syntax for a right join is:

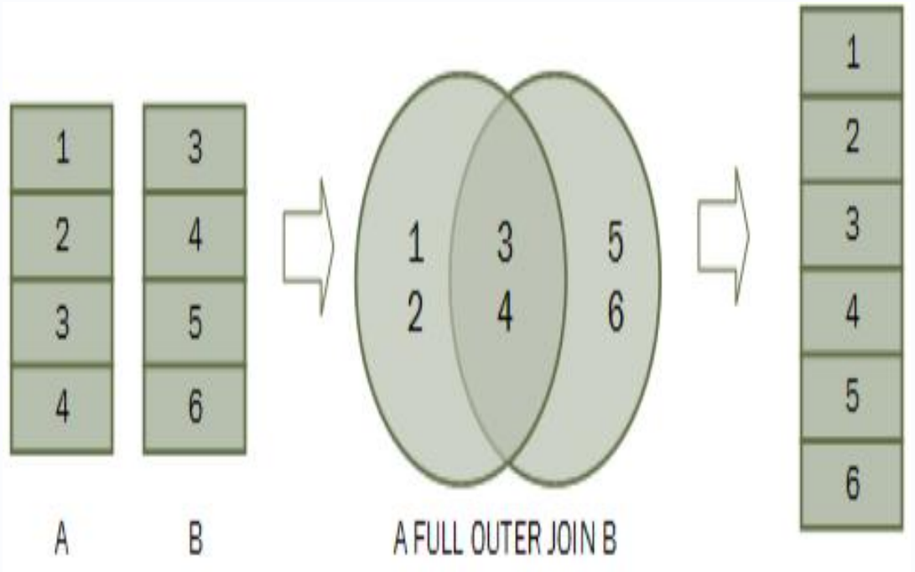
```
SELECT columns
FROM table1
FULL OUTER JOIN table2 ON table1.column = table2.column;
```

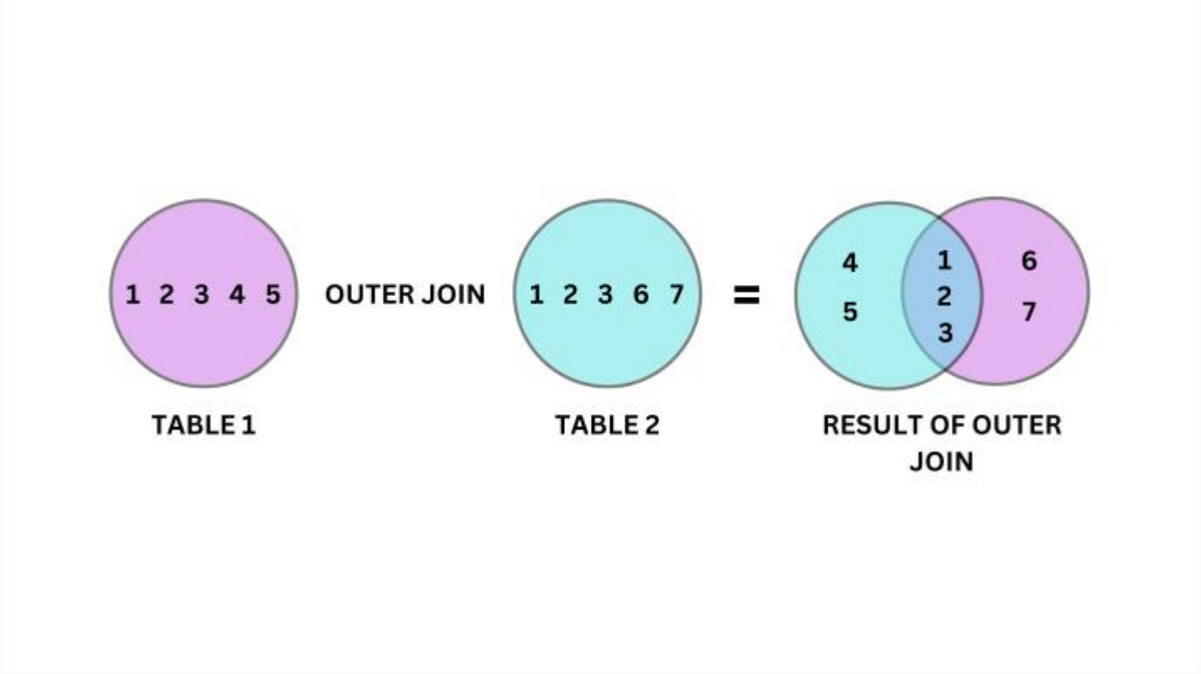
Full outer join is not supported in MySQL, but can be simulated using set operators

Outer Join

Example

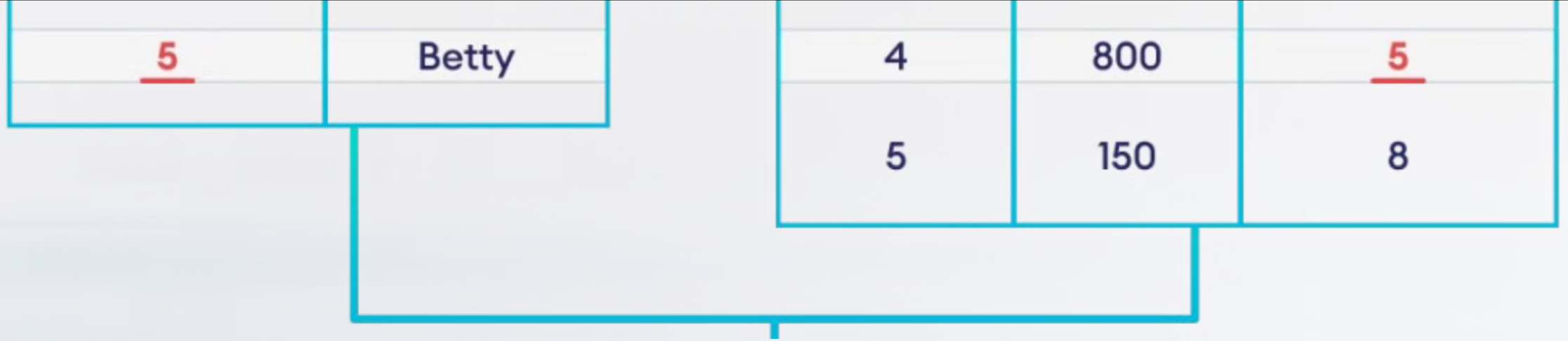
```
SELECT
        column_list
FROM
        A
FULL OUTER JOIN
        B
ON A.Id = B.ID;
```





What is UNION?

UNION is a SQL set operation that allows you to combine the results of two or more SELECT statements into a single result set. It removes any duplicate rows, ensuring that the final result set contains only unique records. This makes UNION particularly useful when you need to combine data from multiple sources or tables, while eliminating redundant information.



Syntax for Union

The syntax for a Union is:

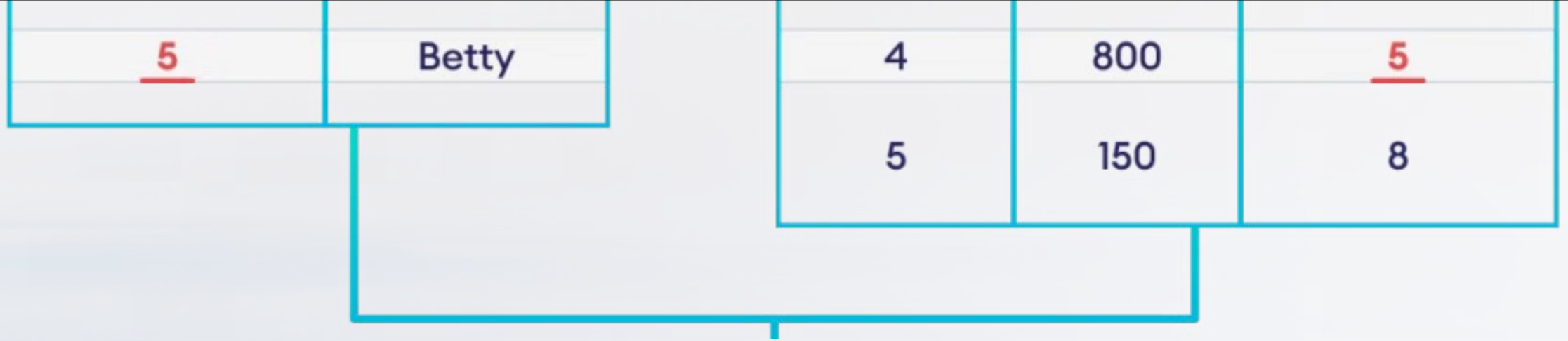
```
SELECT column1, column2, ...
FROM table1
UNION
SELECT column1, column2, ...
FROM table2;
```

Key Points

- The number and order of columns in each SELECT statement must be the same. - The data types of the corresponding columns must be compatible. - UNION removes any duplicate rows from the final result set.

What is UNION ALL?

UNION ALL is another SQL set operation that is similar to UNION, but with one key difference: it does not remove duplicate rows from the final result set. Instead, UNION ALL combines the results of multiple SELECT statements, preserving all rows, even if they are duplicates.



Syntax for Union

The syntax for a Union All is:

```
SELECT column1, column2, ...
FROM table1
UNION ALL
SELECT column1, column2, ...
FROM table2;
```

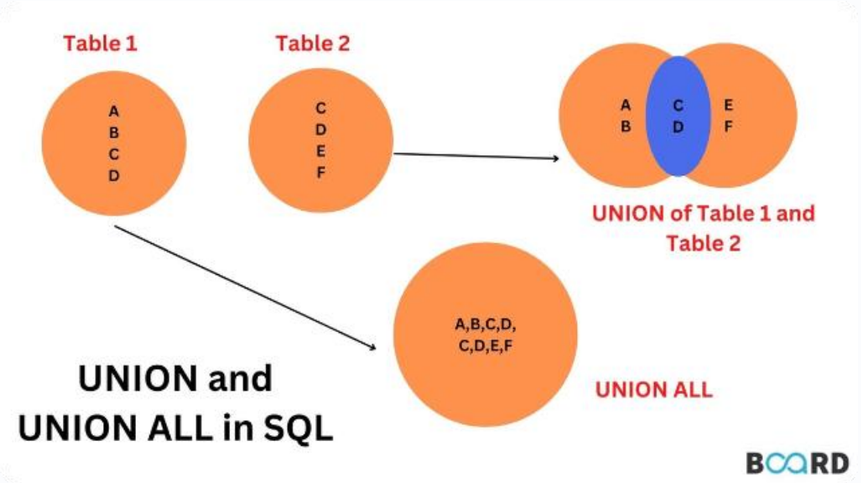
Key Points

- The number and order of columns in each SELECT statement must be the same. - The data types of the corresponding columns must be compatible. - UNION ALL preserves all rows, including any duplicates, in the final result set.

Outer Join

Example

```
SELECT *
FROM Table1
UNION
SELECT *
FROM Table2;
```



Self Join

1

Syntax

SELECT column1, column2, ... FROM table1 t1 JOIN table1 t2 ON t1.column = t2.column;

2

Use Case

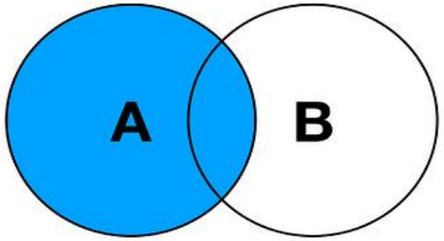
Self joins are used to join a table to itself, based on a related column. This is useful when you need to compare values within the same table, such as finding employees who have the same manager or finding the relationships between different parts of a hierarchical structure.

3

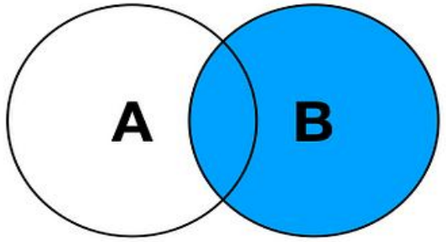
Example

SELECT e1.employee_name AS "Employee", e2.employee_name AS "Manager" FROM employees e1 JOIN employees e2 ON e1.manager_id = e2.employee_id;

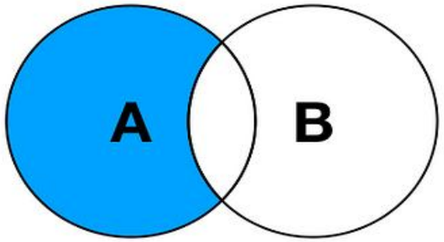
SQL JOINS



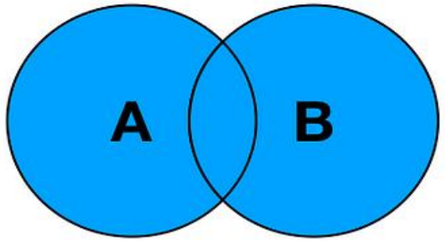
LEFT JOIN



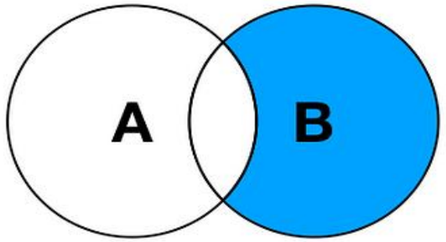
RIGHT JOIN



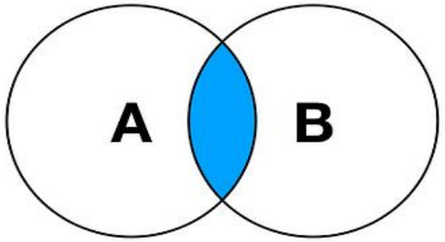
LEFT JOIN EXCLUDING
INNER JOIN



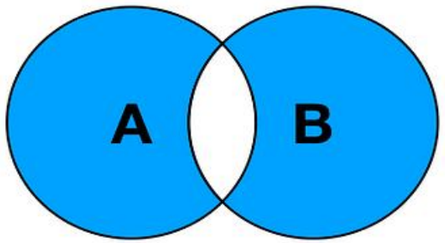
FULL OUTER JOIN



RIGHT JOIN EXCLUDING
INNER JOIN



INNER JOIN



FULL OUTER JOIN EXCLUDING
INNER JOIN