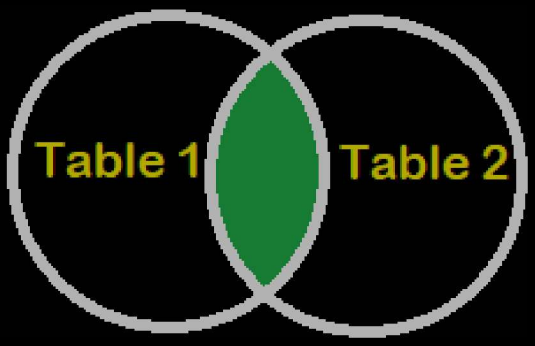
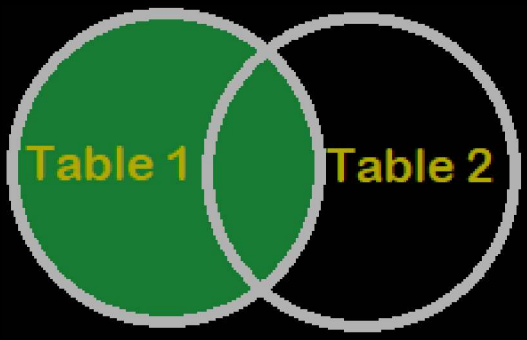
SQL Che

SQL Commands, Functions & Clauses

```
-- Basic SQL commands:

SELECT - retrieves data from a database
FROM - specifies which tables to retrieve data from
WHERE - specifies which rows to retrieve based on certain conditions
GROUP BY - groups rows that have the same values in the specified columns
HAVING - filters groups based on a specified condition
ORDER BY - sorts the retrieved rows in a specified order

-- Aggregate functions:

AVG() - returns the average value of a set of values
COUNT() - returns the number of rows in a table or the number of non-null values
in a column
FIRST() - returns the first value in a set of values
LAST() - returns the last value in a set of values
MAX() - returns the maximum value in a set of values
MIN() - returns the minimum value in a set of values
SUM() - returns the sum of a set of values

-- String functions:

CONCAT() - concatenates two or more strings together
INSTR() - returns the position of a substring within a string
LENGTH() - returns the length of a string
LOWER() - converts a string to lowercase
LTRIM() - removes leading spaces from a string
REPLACE() - replaces all occurrences of a specified string with
another string
```

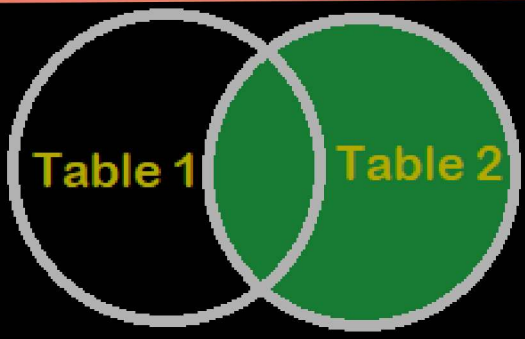



```sql
-- INNER JOIN:

SELECT * FROM table1 INNER JOIN table2 ON table1.
```



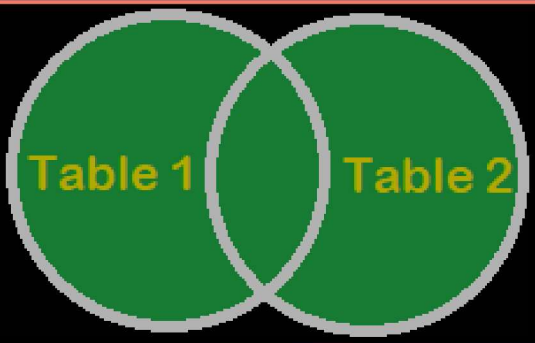
Outer Join

```sql
-- LEFT JOIN:

SELECT * FROM table1 LEFT JOIN table2 ON table1.
```

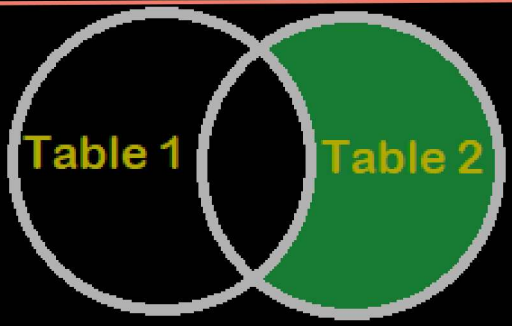


Outer Join

```sql
-- RIGHT JOIN:

SELECT * FROM table1 RIGHT JOIN table2 ON table1.
```

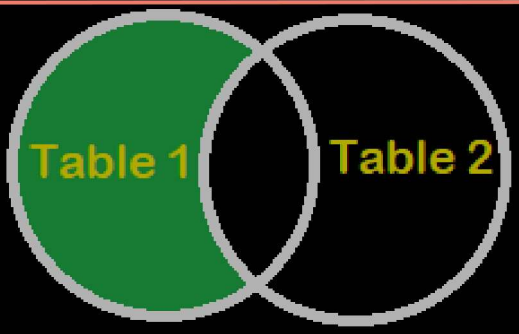



```sql
-- FULL OUTER JOIN:

SELECT * FROM table1 FULL OUTER JOIN table2 ON t
table2.col2
```



Right Outer Join with Exclusion

```sql
SELECT *
FROM table1
RIGHT JOIN table2 ON table1.col1 = table2.col2
WHERE table1.col1 IS NULL;
```



Left Outer Join with Exclusion

```sql
SELECT *
FROM table1
LEFT JOIN table2 ON table1.col1 = table2.col2
WHERE table2.col1 IS NULL;
```

Full Outer Join with Exclusion

SQL Examples (Easy to Advanced)

```
-- Sample data for the "employees" table:

id | name   | department | salary | hired_on
-----------------------------------------------
1  | Alice | HR         | 55000  | 2020-01-01
2  | Bob    | Marketing  | 65000  | 2020-02-01
3  | Charlie| IT          | 75000  | 2020-03-01
4  | Dave   | Sales      | 80000  | 2020-04-01
5  | Eve    | HR         | 60000  | 2020-05-01

-- Sample data for the "sales" table:

id | employee_id | product    | sale_date       | sale_amount
-------------------------------------------------------------
1  | 1           | Widget     | 2020-06-01      | 1000
2  | 1           | Gadget     | 2020-07-01      | 2000
3  | 2           | Widget     | 2020-08-01      | 3000
4  | 2           | Gadget     | 2020-09-01      | 4000
5  | 2           | Thingamajig| 2020-10-01      | 5000
6  | 3           | Thingamajig| 2020-11-01      | 6000
7  | 4           | Widget     | 2020-12-01      | 7000
8  | 4           | Gadget     | 2021-01-01      | 8000
9  | 4           | Thingamajig| 2021-02-01      | 9000
10 | 5           | Widget     | 2021-03-01      | 10000

-- Sample data for the "products" table:

id | product    | price
-----------------------
1  | Widget     | 100
2  | Gadget     | 200
```

Some Advanced Topics in SQL

1-Recursive queries: These are queries that can reference themselves in order to perform a certain action, such as querying hierarchical data.

```
-- Recursive queries:

WITH RECURSIVE cte_name AS (
    SELECT ...
    UNION [ALL]
    SELECT ...
    FROM cte_name
    WHERE ...
)
SELECT ...
FROM cte_name;
```

2-Window functions: These are functions that perform a calculation over a set of rows, similar to an aggregate function, but return a value for each row in the result set.

```
-- Window functions:

SELECT col1, col2, function_name(col3) OVER (PARTITION BY col1 ORDER BY col2) as col4
FROM table_name;
```

3-Common table expressions (CTEs): These are named temporary result sets that can be used within a SELECT, INSERT, UPDATE, DELETE, or CREATE VIEW statement. They are often used to simplify complex queries by breaking them up into smaller, more manageable pieces.

```
-- Common table expressions (CTEs):

WITH cte_name AS (
    SELECT ...
```