Database Using SQL Server & Raw Data available over my GitHub Profile

MySQL

PostgreSQL

JOINS With 12 Most Used Queries

AnshLibrary

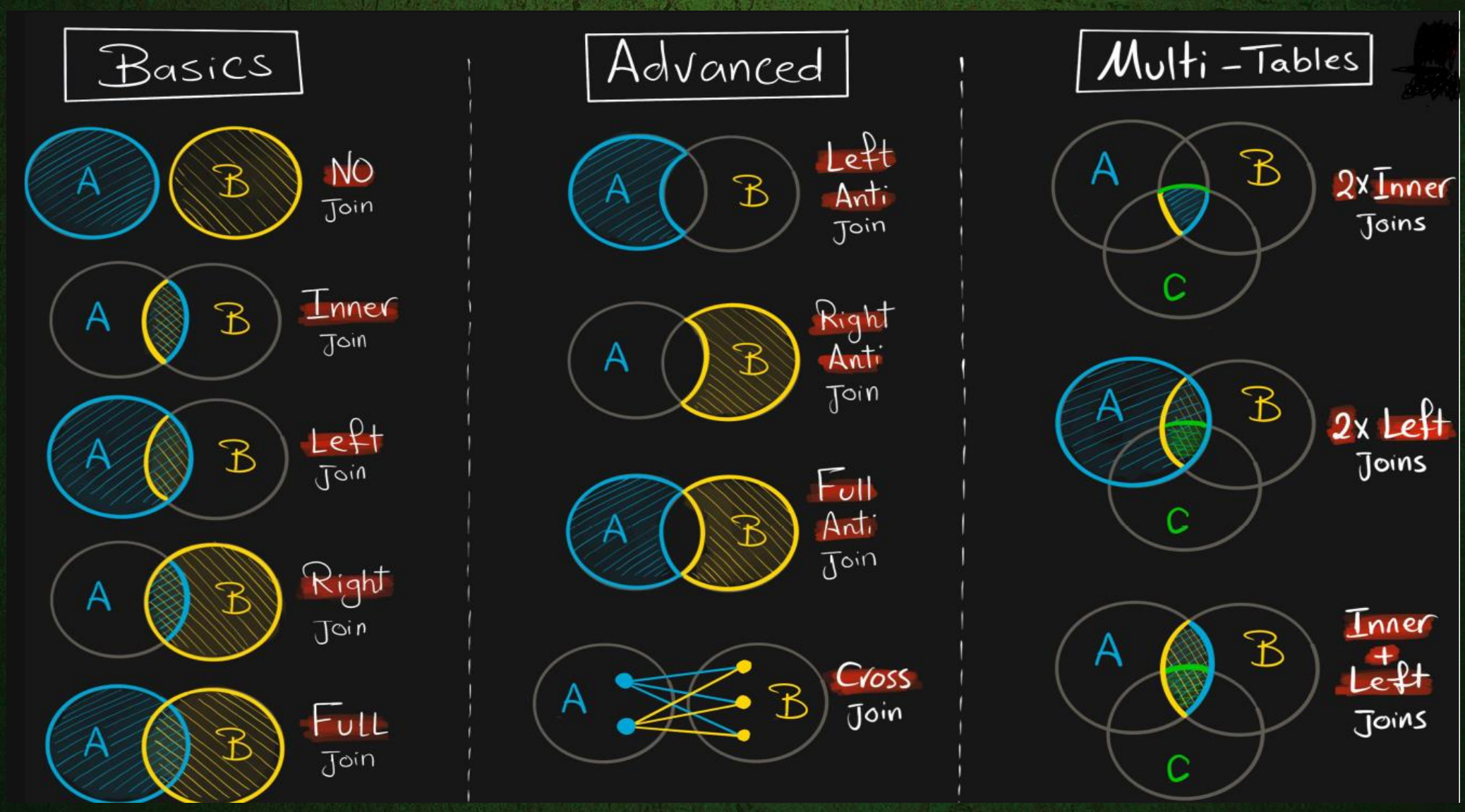
What is JOIN

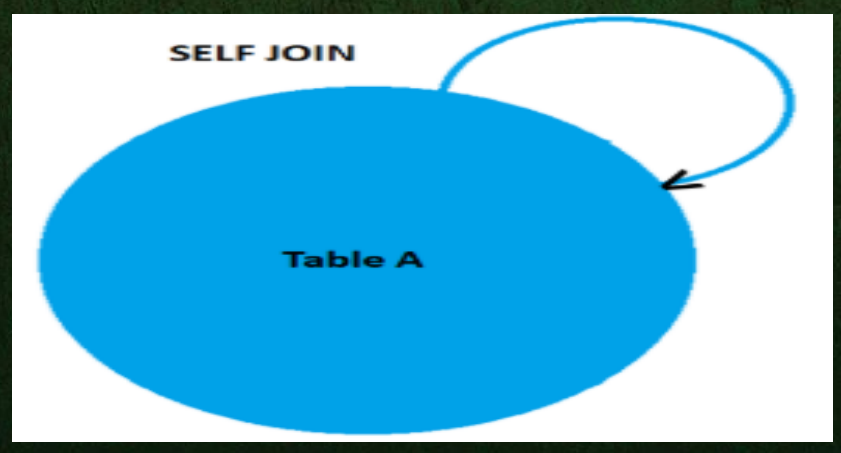
In SQL Server, a join is a way to combine data from two or more tables based on a related column between them. The join operation allows you to retrieve data from multiple tables at once and create a new virtual table that contains information from all the tables involved in the join.

Syntax of VIEW:

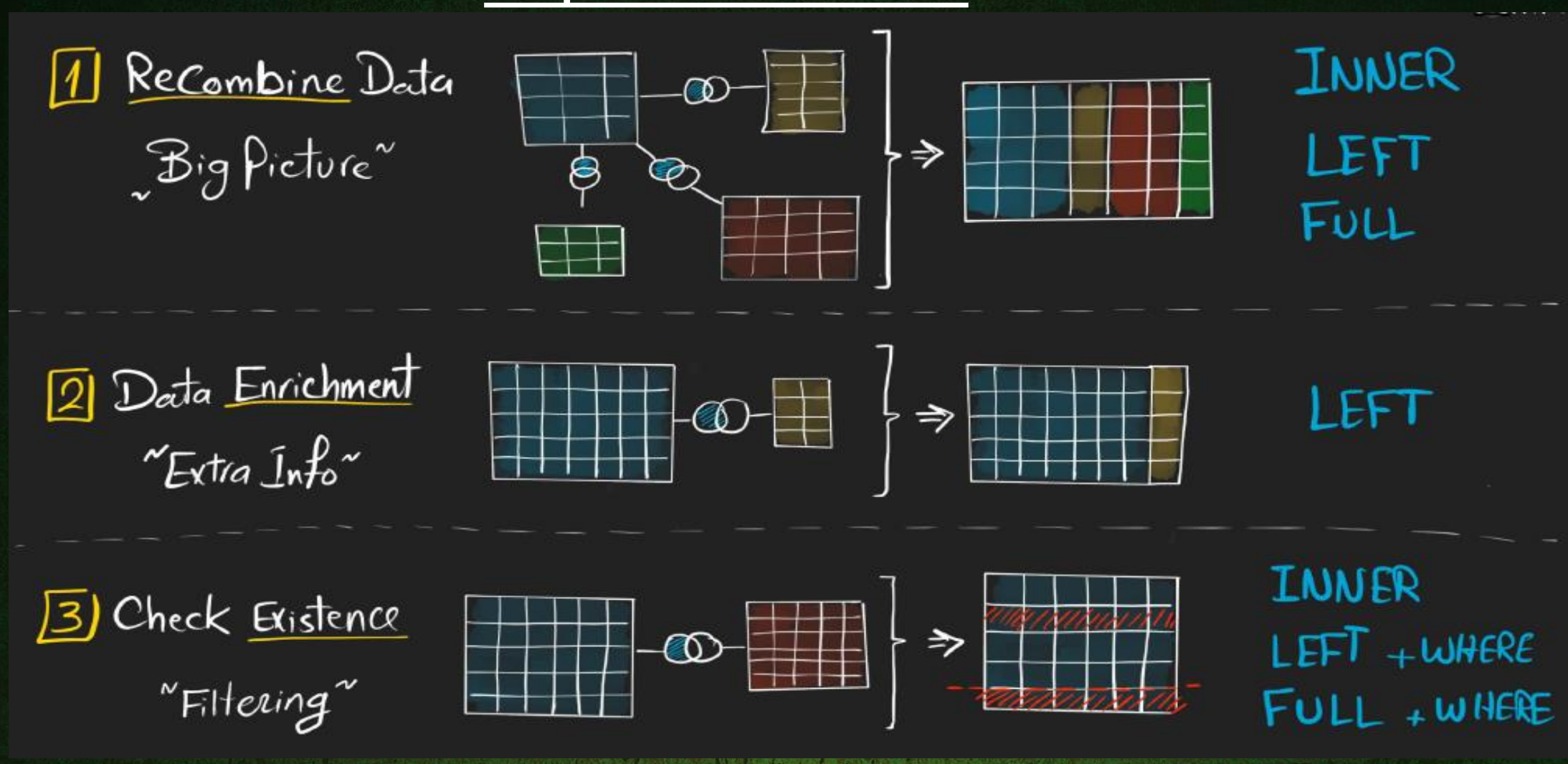
```
SELECT column_list FROM table1 JOIN table2
ON table1.column = table2.column;
```

Types Of Joins:

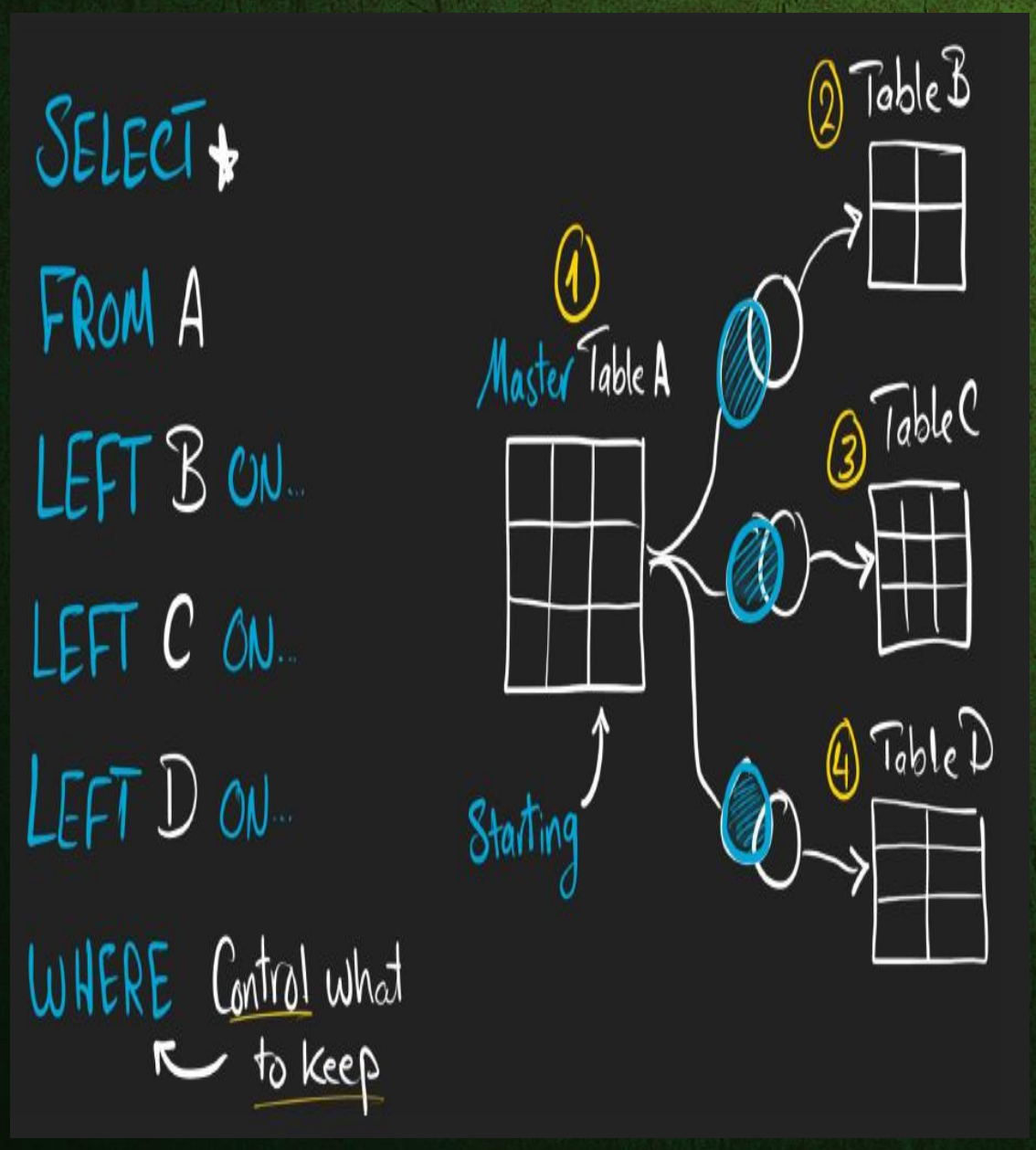


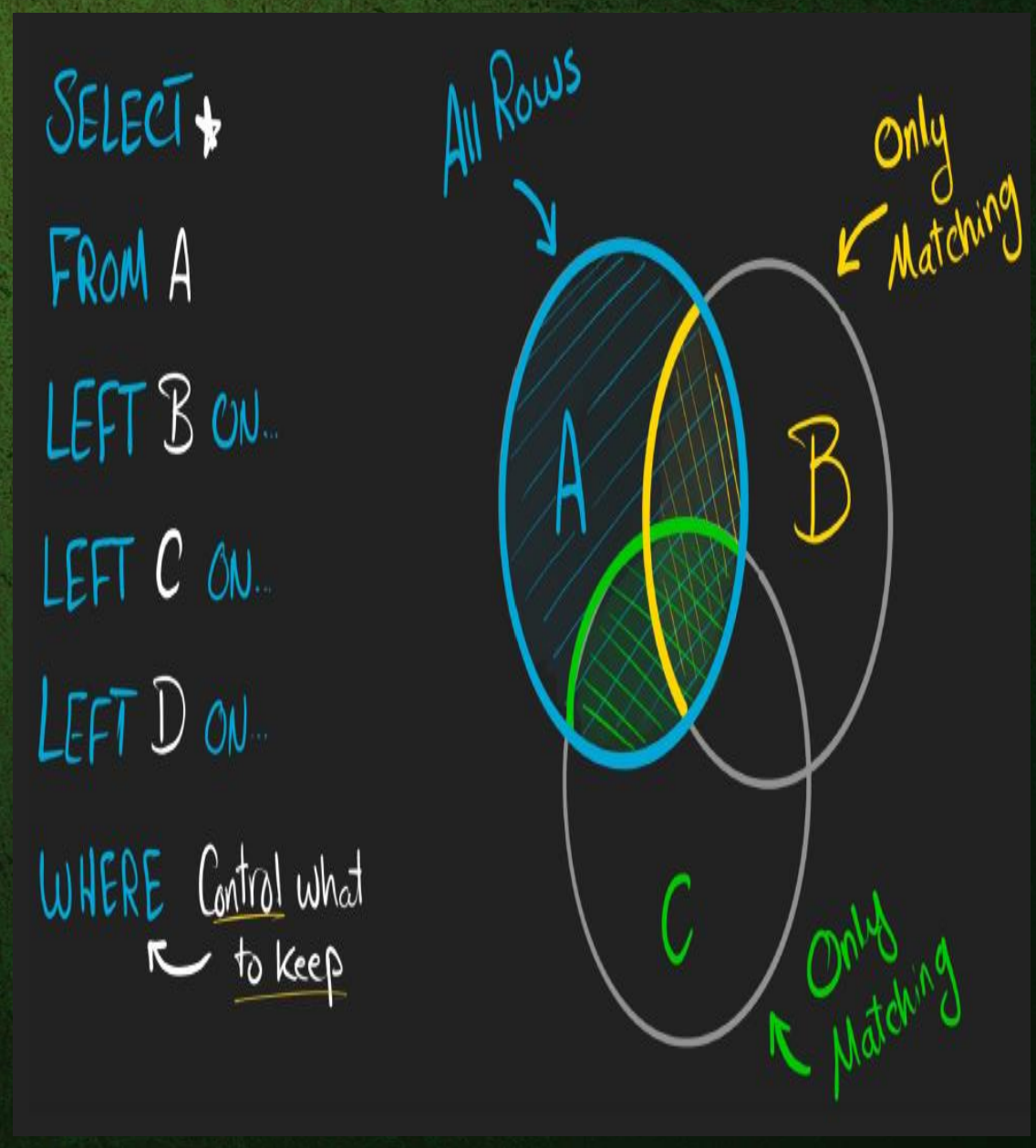


Purpose to Use Joins:



How I Join Multiple Tables





01

Get all customers who haven’t place any orders

```
SELECT *
FROM
Customers AS c
LEFT JOIN orders AS o
ON c.id = o.customer_id
WHERE o.customer_id IS NULL
```

Results | Messages

	Customer_id	First_Name	Last_Name	Country	Score	OrderID	ProductID	CustomerID	SalesPersonID	OrderDate	ShipDate	OrderStatus	ShipAddress	BillAddress	Quantity	Sales	CreationTime
1	7	John	Doe	USA	680	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
2	8	Sophia	Meier	Germany	540	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
3	10	Ava	NULL	UK	770	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
4	11	Liam	O'Brien	Ireland	790	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
5	12	Emma	Nguyen	Vietnam	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
6	13	David	Lee	South Korea	810	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
7	14	Isabella	Rossi	Italy	620	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL
8	15	Noah	Patel	India	600	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

02

Customer Purchase Rank in Country

```
SELECT

    c.Country,
    c.Customer_id,
    c.First_Name,
    SUM(o.Sales) AS TotalSpent,
     RANK() OVER(PARTITION BY c.Country ORDER BY SUM(o.Sales)
DESC) AS CountryRank

FROM Orders o
JOIN Customers c ON o.CustomerID = c.Customer_id
GROUP BY c.Country, c.Customer_id, c.First_Name;
```

Results | Messages

	Country	Customer_id	First_Name	TotalSpent	CountryRank
1	Canada	6	Emily	180.00	1
2	Germany	4	Mark	840.00	1
3	Germany	1	Jossef	605.00	2
4	Mexico	9	Carlos	280.00	1
5	USA	3	Mary	679.00	1
6	USA	5	Anna	347.00	2
7	USA	2	Kevin	274.00	3

Use by: Rank customers within their country based on total sales.

03

Latest Order Per Customer Using CTE + Window

```
WITH RankedOrders AS (
    SELECT *,
        ROW_NUMBER() OVER(PARTITION BY CustomerID
ORDER BY OrderDate DESC) AS rn
    FROM Orders
)
SELECT *
FROM RankedOrders
WHERE rn = 1;
```

Results | Messages

	OrderID	ProductID	CustomerID	SalesPersonID	OrderDate	ShipDate	OrderStatus	ShipAddress	BillAddress	Quantity	Sales	CreationTime	rn
1	27	101	1	3	2025-07-17	2025-07-21	On Hold	6464 Crystal River	7779 Old Hollow Dr	8	220.00	2025-07-17 16:05:15.0000000	1
2	22	106	2	3	2025-07-12	2025-07-16	Shipped	3333 Violet Blvd	6777 Oak Hollow Dr	3	120.00	2025-07-12 11:20:30.0000000	1
3	29	104	3	5	2025-07-19	2025-07-23	Cancelled	2330 Stonegate Way	NULL	10	260.00	2025-07-19 18:25:35.0000000	1
4	28	103	4	2	2025-07-18	2025-07-22	Packed	3922 Amber Crest	1100 Jade Hill Blvd	9	240.00	2025-07-18 17:15:25.0000000	1
5	24	107	5	4	2025-07-14	2025-07-18	Shipped	1111 Breeze Dr	3000 Hill Valley Rd	5	160.00	2025-07-14 13:40:50.0000000	1
6	25	107	6	2	2025-07-15	2025-07-19	Delivered	5000 West Palm Ln	1044 Sunny Dr	6	180.00	2025-07-15 14:51:55.0000000	1
7	30	105	9	4	2025-07-20	2025-07-24	Processing	NULL	5599 Ridgeview Ln	11	280.00	2025-07-20 19:35:45.0000000	1

Use by: Identify the most recent order placed by each customer.

04

Category-Wise Sales Contribution

```
SELECT

    p.Category,
    SUM(o.Sales) AS CategorySales,
    SUM(o.Sales) * 100.0 / SUM(SUM(o.Sales)) OVER() AS
    SalesPercent

    FROM Orders o
    JOIN Products p ON o.ProductID = p.ProductID
    GROUP BY p.Category;
```

	Category	CategorySales	SalesPercent
1	Accessories	1633.00	50.951638
2	Clothing	1572.00	49.048361

Use by: Break down product category sales and show percentage contribution.

05

Top Salesperson in Each Department

```
WITH DeptSales AS (
    SELECT
        e.Department,
        e.FirstName,
       SUM(o.Sales) AS TotalSales,
        RANK() OVER(PARTITION BY e.Department ORDER BY
SUM(o.Sales) DESC) AS rnk

    FROM Employees e
    JOIN Orders o ON e.Employee_id = o.SalesPersonID
    GROUP BY e.Department, e.FirstName
)
SELECT * FROM DeptSales WHERE rnk = 1;
```

Results | Messages

	Department	FirstName	TotalSales	rnk
1	Marketing	Kevin	919.00	1
2	Sales	Michael	767.00	1

Use by: Use CTE + Rank to find top seller per department.

06

Sales Conversion Rate by Employee

```
SELECT
    e.Employee_id,
    e.FirstName,
    COUNT(o.OrderID) AS TotalOrders,
    COUNT(CASE WHEN o.OrderStatus = 'Delivered' THEN 1 END) AS SuccessfulDeliveries,

--NULLIF(COUNT(o.OrderID), 0) returns NULL instead of 0.
--Division by NULL is safe — it just results in NULL, avoiding the runtime error.

    100.0 * COUNT(CASE WHEN o.OrderStatus = 'Delivered' THEN 1 END)
        / NULLIF(COUNT(o.OrderID), 0) AS ConversionRate,

--If you prefer to show 0% instead of NULL, you can wrap it with ISNULL() or COALESCE():

    COALESCE(100.0 * COUNT(CASE WHEN o.OrderStatus = 'Delivered' THEN 1 END)
        / NULLIF(COUNT(o.OrderID), 0), 0) AS ConversionRate2,

--Cast(…As DECIMAL (5,2)) directly controls decimal precision (2 digits after the decimal)

   CAST( COALESCE(100.0 * COUNT(CASE WHEN o.OrderStatus = 'Delivered' THEN 1 END)
        / NULLIF(COUNT(o.OrderID), 0),0) AS DECIMAL(5,2)) AS ConversionRate

FROM Employees e
LEFT JOIN Orders o ON e.Employee_id = o.SalesPersonID
GROUP BY e.Employee_id, e.FirstName;
```

	Employee_id	FirstName	TotalOrders	SuccessfulDeliveries	ConversionRate	ConversionRate2	ConversionRate
1	1	Frank	4	1	25.000000000000	25.000000000000	25.00
2	2	Kevin	7	2	28.571428571428	28.571428571428	28.57
3	3	Mary	7	1	14.285714285714	14.285714285714	14.29
4	4	Michael	5	1	20.000000000000	20.000000000000	20.00
5	5	Carol	7	3	42.857142857142	42.857142857142	42.86
6	6	Emily	0	0	NULL	0.000000000000	0.00
7	7	James	0	0	NULL	0.000000000000	0.00
8	8	Olivia	0	0	NULL	0.000000000000	0.00
9	9	Ethan	0	0	NULL	0.000000000000	0.00
10	10	Sophia	0	0	NULL	0.000000000000	0.00
11	11	Daniel	0	0	NULL	0.000000000000	0.00
12	12	Ava	0	0	NULL	0.000000000000	0.00
13	13	William	0	0	NULL	0.000000000000	0.00
14	14	Isabella	0	0	NULL	0.000000000000	0.00
15	15	Liam	0	0	NULL	0.000000000000	0.00

Use by: Total orders handled vs. completed deliveries.

AnshLibrary

07

Products Never Ordered (LEFT JOIN Anti-Join)

```
SELECT

p.ProductID,
p.Product
FROM Products p
LEFT JOIN Orders o ON p.ProductID = o.ProductID
WHERE o.ProductID IS NULL;
```

Results | Messages

	ProductID	Product
1	108	Jersey
2	109	Shorts
3	110	Sunglasses
4	111	Backpack
5	112	Rain Jacket
6	113	Arm Warmers
7	114	Chain Lube
8	115	Multi-tool
9	116	Bike Pump
10	117	Pedals
11	118	Waterproof Pants
12	119	Repair Kit
13	120	Reflective Vest

08

Total Revenue by Region (Derived Table Join)

```
SELECT

c.Country,
SUM(o.Sales) AS TotalSales
FROM Orders o
JOIN Customers c
ON o.CustomerID = c.Customer_id
GROUP BY c.Country;
```

Results | Messages

	Country	TotalSales
1	Canada	180.00
2	Germany	1445.00
3	Mexico	280.00
4	USA	1300.00

09

Order History Audit from Both Tables

```
SELECT

 o.OrderID,
o.OrderDate,
oa.OrderDate AS ArchivedOrderDate,
o.Sales,
oa.Sales AS ArchivedSales

FROM Orders o
LEFT JOIN OrdersArchive oa
ON o.OrderID = oa.OrderID
AND o.CustomerID = oa.CustomerID;
```

Results | Messages

	OrderID	OrderDate	ArchivedOrderDate	Sales	ArchivedSales
4	4	2025-01-20	2024-04-20	60.00	60.00
5	4	2025-01-20	2024-04-20	60.00	60.00
6	5	2025-02-01	2024-05-01	25.00	25.00
7	6	2025-02-05	2024-05-05	50.00	50.00
8	6	2025-02-05	2024-05-05	50.00	50.00
9	6	2025-02-05	2024-05-05	50.00	50.00
10	7	2025-02-15	NULL	30.00	NULL
11	8	2025-02-18	2024-06-18	90.00	45.00
12	9	2025-03-10	2024-06-20	20.00	25.00
13	10	2025-03-15	NULL	60.00	NULL
14	11	2025-07-01	NULL	44.00	NULL
15	12	2025-07-02	NULL	55.00	NULL
16	13	2025-07-03	NULL	66.00	NULL
17	14	2025-07-04	NULL	77.00	NULL
18	15	2025-07-05	NULL	88.00	NULL
19	16	2025-07-06	NULL	99.00	NULL
20	17	2025-07-07	NULL	110.00	NULL
21	18	2025-07-08	NULL	121.00	NULL
22	19	2025-07-09	NULL	132.00	NULL
23	20	2025-07-10	NULL	143.00	NULL
24	21	2025-07-11	NULL	90.00	NULL
25	22	2025-07-12	NULL	120.00	NULL
26	23	2025-07-13	NULL	140.00	NULL
27	24	2025-07-14	NULL	160.00	NULL
28	25	2025-07-15	NULL	180.00	NULL

Use by: Match and compare current and archived records.

10

Customers with multiple orders & compare first vs last

```
WITH RankedOrders AS (
  SELECT *,

   ROW_NUMBER() OVER(PARTITION BY CustomerID ORDER BY
OrderDate ASC) AS FirstOrder,
   ROW_NUMBER() OVER(PARTITION BY CustomerID ORDER BY
OrderDate DESC) AS LastOrder

  FROM Orders
)
SELECT
First.CustomerID,
First.OrderDate AS FirstOrderDate,
Last.OrderDate AS LastOrderDate

FROM RankedOrders First
JOIN RankedOrders Last
ON First.CustomerID = Last.CustomerID
WHERE First.FirstOrder = 1 AND Last.LastOrder = 1;
```

	CustomerID	FirstOrderDate	LastOrderDate
1	1	2025-01-10	2025-07-17
2	2	2025-01-01	2025-07-12
3	3	2025-01-05	2025-07-19
4	4	2025-02-18	2025-07-18
5	5	2025-07-04	2025-07-14
6	6	2025-07-15	2025-07-15
7	9	2025-07-20	2025-07-20

Use by: SELF JOIN + Window Function

11

Products never ordered

```
SELECT

p.ProductID,
p.Product

FROM Products p
WHERE NOT EXISTS (
SELECT 1 FROM Orders o WHERE
   o.ProductID = p.ProductID
);
```

Results | Messages

	ProductID	Product
1	108	Jersey
2	109	Shorts
3	110	Sunglasses
4	111	Backpack
5	112	Rain Jacket
6	113	Arm Warmers
7	114	Chain Lube
8	115	Multi-tool
9	116	Bike Pump
10	117	Pedals
11	118	Waterproof Pants
12	119	Repair Kit
13	120	Reflective Vest

Use by: ANTI JOIN (NOT EXISTS)

AnshLibrary

12

Orders With Missing Billing or Shipping Info View

```
SELECT

e.Employee_id,
e.FirstName

FROM Employees e
WHERE NOT EXISTS (
   SELECT 1 FROM Orders o
      WHERE o.SalesPersonID = e.Employee_id
      AND YEAR(o.OrderDate) = 2025
);
```

Results | Messages

	Employee_id	First Name
1	6	Emily
2	7	James
3	8	Olivia
4	9	Ethan
5	10	Sophia
6	11	Daniel
7	12	Ava
8	13	William
9	14	Isabella
10	15	Liam

Use by: ANTI JOIN