

JOINS & UNIONS IN SQL

JOINS & UNIONS IN SQL

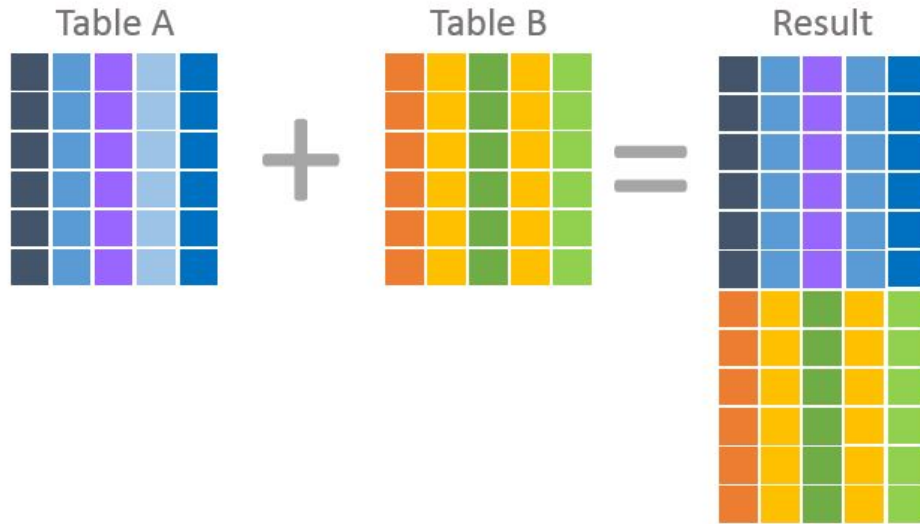
LEARNING OBJECTIVES

1. Learn the SQL tools for appending similar data together.
2. Explore combining data from different tables together.
3. Use the SQL commands **JOIN** and **UNION** to answer data questions.
4. Introduce the SQL structure to **JOIN** data from multiple sources.

JOINS & UNIONS IN SQL

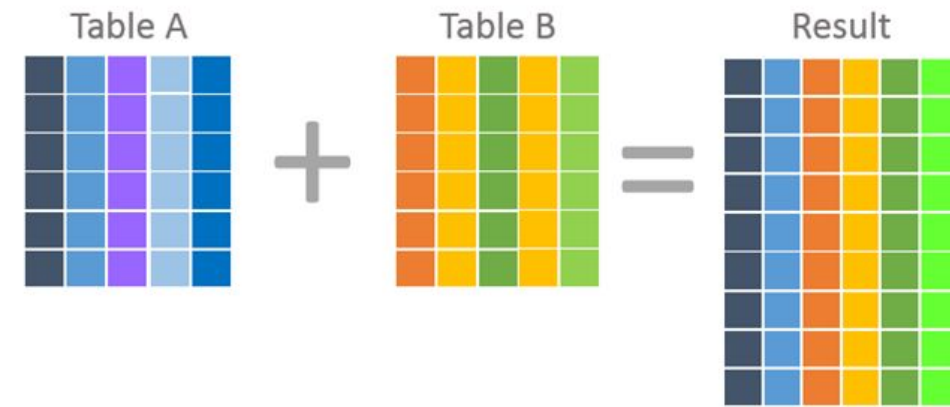
INTRODUCTION: UNIONS AND JOINS

UNIONS AND JOINS: ILLUSTRATION



A **UNION** merges rows of *similar* data to create a new set.

A **JOIN** combines columns from tables using common unique identifiers (keys).



UNIONS AND JOINS: COMPARING TABLES



ACTIVITY

DIRECTIONS

1. Navigate to the tables labeled FY17 and FY18.
2. Observe the likely meaning of each column.
3. Contrast the data type structures for both tables.
4. Make a list of similarities and differences.

DELIVERABLE

Share your observations on how this data might be combined to create meaningful business reports.

JOINS & UNIONS IN SQL

DEMO: UNIONS

UNIONS: SAMPLE CODE STRUCTURE

Let's look at some simple mock syntax for a **UNION**:

```
SELECT field1
  FROM table1
 WHERE field1= X
UNION
SELECT Field1
  FROM table2
 WHERE field1= X
ORDER BY 1
```

JOINS & UNIONS IN SQL

GUIDED PRACTICE: UNIONS

THREE TYPES OF UNIONS: MERGING TWO TABLES

We'll build three different examples of UNIONS:

- A table UNION.
- A column UNION.
- A combination of the two.

We'll be using the following sample HR tables for illustration:

Employees1

id	first_name	last_name	current_salary
2	Gabe	Moore	50000
3	Doreen	Mandeville	60000
5	Simone	MacDonald	55000

Employees2

id	first_name	last_name	current_salary
7	Madisen	Flateman	75000
11	Ian	Paasche	120000
13	Mimi	St. Felix	70000

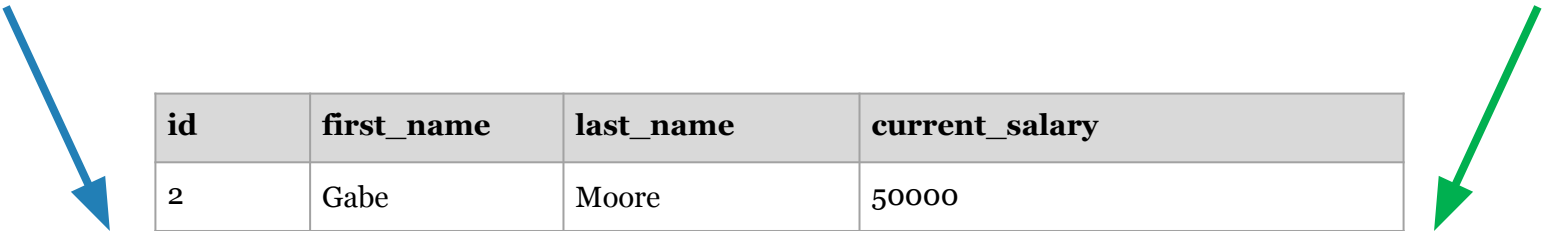
THREE TYPES OF UNIONS: MERGING TWO TABLES

A **table** UNION includes selections from different tables:

```
SELECT * FROM Employees1 UNION SELECT * FROM Employees2
```

id	first_name	last_name	current_salary
2	Gabe	Moore	50000
3	Doreen	Mandeville	60000
5	Simone	MacDonald	55000

id	first_name	last_name	current_salary
7	Madisen	Flateman	75000
11	Ian	Paasche	120000
13	Mimi	St. Felix	70000



id	first_name	last_name	current_salary
2	Gabe	Moore	50000
3	Doreen	Mandeville	60000
5	Simone	MacDonald	55000
7	Madisen	Flateman	75000
11	Ian	Paasche	120000
13	Mimi	St. Felix	70000

THREE TYPES OF UNIONS: MERGING TWO TABLES

A **column** UNION is when you UNION columns from the same table together:

```
SELECT first_name FROM Employees1  
UNION  
SELECT last_name FROM Employees1
```

id	first_name	last_name	current_salary
2	Gabe	Moore	50000
3	Doreen	Mandeville	60000
5	Simone	MacDonald	55000



id	first_name
2	Gabe
3	Doreen
5	Simone
2	Moore
3	Mandeville
5	MacDonald

THREE TYPES OF UNIONS: MERGING TWO TABLES

A combination of *column* and *table* **unions** brings together specific columns from two different tables:

```
SELECT first_name FROM Employees1
UNION
SELECT last_name FROM Employees1
UNION
SELECT first_name FROM Employees2
UNION
SELECT last_name FROM Employees2
```

id	first_name
2	Gabe
3	Doreen
5	Simone
2	Moore
3	Mandeville
5	MacDonald
7	Madisen
11	Ian
13	Mimi
7	Flateman
11	Paasche
13	St. Felix

THREE TYPES OF UNIONS: MERGING TWO TABLES

Let's try this with our PostgreSQL database. Create a table UNION:

```
SELECT *  
FROM FY17
```

UNION

```
SELECT *  
FROM FY18
```

	fy numeric	pd numeric	store_na... character	week1 numeric	week2 numeric	week3 numeric	week4 numeric
<input type="checkbox"/>	18	2	SAN DIEGO	754.959568	803.757709	5687.76688	219.527173
<input type="checkbox"/>	18	2	DALLAS	103.293083	217.776958	648.601179	4531.9008
<input type="checkbox"/>	18	2	SEATTLE	869.439661	595.887082	954.010546	746.064601
<input type="checkbox"/>	17	2	SAN DIEGO	000.715479	741.061154	400.657862	112.872637
<input type="checkbox"/>	17	2	DALLAS	266.472034	698.658564	045.834247	762.222799
<input type="checkbox"/>	17	2	PORTLAND	569.060961	415.029643	67.8186051	72.2430083
<input type="checkbox"/>	17	2	PORTLAND	351.793961	485.721046	425.021263	306.720805
<input type="checkbox"/>	18	2	PORTLAND	729.988537	06.4388102	150.983333	198.229664
<input type="checkbox"/>	18	2	PORTLAND	301.496764	3956.17356	127.618963	779.973338
<input type="checkbox"/>	18	2	SEATTLE	789.640585	194.083103	282.101838	744.869032
<input type="checkbox"/>	18	2	PORTLAND	002.225137	125.229579	245.442398	7059.23758
<input type="checkbox"/>	17	2	SEATTLE	692.497933	181.314618	641.401526	082.684282
<input type="checkbox"/>	17	2	PHOENIX	72.3659485	900.969922	737.209742	471.024603
<input type="checkbox"/>	18	2	VANCOUV...	913.520693	583.003705	768.463565	214.983293
<input type="checkbox"/>	17	2	VANCOUV...	293.769182	447.159002	092.626357	025.499113

THREE TYPES OF UNIONS: MERGING TWO TABLES

Next, create a **column** UNION :

```
SELECT FY, PD, STORE_NAME, WEEK1
FROM FY17
UNION
SELECT FY, PD, STORE_NAME, WEEK2
FROM FY17
```

	fy numeric	pd numeric	store_na... character	week1 numeric
<input type="checkbox"/>	17	2	SEATTLE	692.497933
<input type="checkbox"/>	17	2	SEATTLE	431.070306
<input type="checkbox"/>	17	2	PORTLAND	351.793961
<input type="checkbox"/>	17	2	PORTLAND	569.060961
<input type="checkbox"/>	17	2	PORTLAND	943.511757
<input type="checkbox"/>	17	2	VANCOUV...	293.769182
<input type="checkbox"/>	17	2	SAN FRA...	888.715186
<input type="checkbox"/>	17	2	SAN DIEGO	558.010244
<input type="checkbox"/>	17	2	SAN DIEGO	000.715479
<input type="checkbox"/>	17	2	DALLAS	266.472034
<input type="checkbox"/>	17	2	PHOENIX	72.3659485

UNION

fy numeric	pd numeric	store_na... character	week2 numeric
17	2	SEATTLE	181.314618
17	2	SEATTLE	53.2214114
17	2	PORTLAND	485.721046
17	2	PORTLAND	415.029643
17	2	PORTLAND	118.030799
17	2	VANCOUV...	447.159002
17	2	SAN FRA...	917.598135
17	2	SAN DIEGO	760.342741
17	2	SAN DIEGO	741.061154
17	2	DALLAS	698.658564
17	2	PHOENIX	900.969922



fy numeric	pd numeric	store_na... character	week1 numeric
17	2	VANCOUV...	293.769182
17	2	SEATTLE	181.314618
17	2	DALLAS	266.472034
17	2	VANCOUV...	447.159002
17	2	PHOENIX	900.969922
17	2	PORTLAND	569.060961
17	2	PORTLAND	118.030799
17	2	PORTLAND	943.511757
17	2	SAN DIEGO	760.342741
17	2	PORTLAND	485.721046
17	2	PORTLAND	351.793961
17	2	DALLAS	698.658564
17	2	SAN FRA...	917.598135

THREE TYPES OF UNIONS: MERGING TWO TABLES

A combination of column and table UNIONS:

```
SELECT FY, PD, STORE_NAME, WEEK1 FROM FY17
UNION
SELECT FY, PD, STORE_NAME, WEEK2 FROM FY17
UNION
SELECT FY, PD, STORE_NAME, WEEK3 FROM FY17
UNION
SELECT FY, PD, STORE_NAME, WEEK4 FROM FY17
UNION
SELECT FY, PD, STORE_NAME, WEEK1 FROM FY18
UNION
SELECT FY, PD, STORE_NAME, WEEK2 FROM FY18
UNION
SELECT FY, PD, STORE_NAME, WEEK3 FROM FY18
UNION
SELECT FY, PD, STORE_NAME, WEEK4 FROM FY18
ORDER BY 1,2,3
```

fy numeric	pd numeric	store_na... character	week1 numeric
17	2	SEATTLE	181.314618
17	2	SEATTLE	07.5352884
17	2	SEATTLE	641.401526
17	2	SEATTLE	53.2214114
17	2	SEATTLE	082.684282
17	2	SEATTLE	085.558181
17	2	VANCOUV...	293.769182
17	2	VANCOUV...	447.159002
17	2	VANCOUV...	025.499113
17	2	VANCOUV...	092.626357
18	2	DALLAS	4531.9008
18	2	DALLAS	103.293083
18	2	DALLAS	648.601179
18	2	DALLAS	217.776958
18	2	PHOENIX	349.958989
18	2	PHOENIX	769.625787
18	2	PHOENIX	318.069613
18	2	PHOENIX	605.303381
18	2	PORTLAND	301.496764

UNION RULES

Remember these **four** rules when using **UNION**:

1. You must match the number of columns and they must be of compatible data types.
2. You can only have one **ORDER BY** at the bottom of your full SELECT statement.
3. **UNION** removes **exact** duplicates; **UNION ALL** allows duplicates.
4. Conditions between **UNION SELECT** statements should match.

Pro Tip: While you may find yourself altering these rules in practice, keep in mind that this may damage your data integrity.

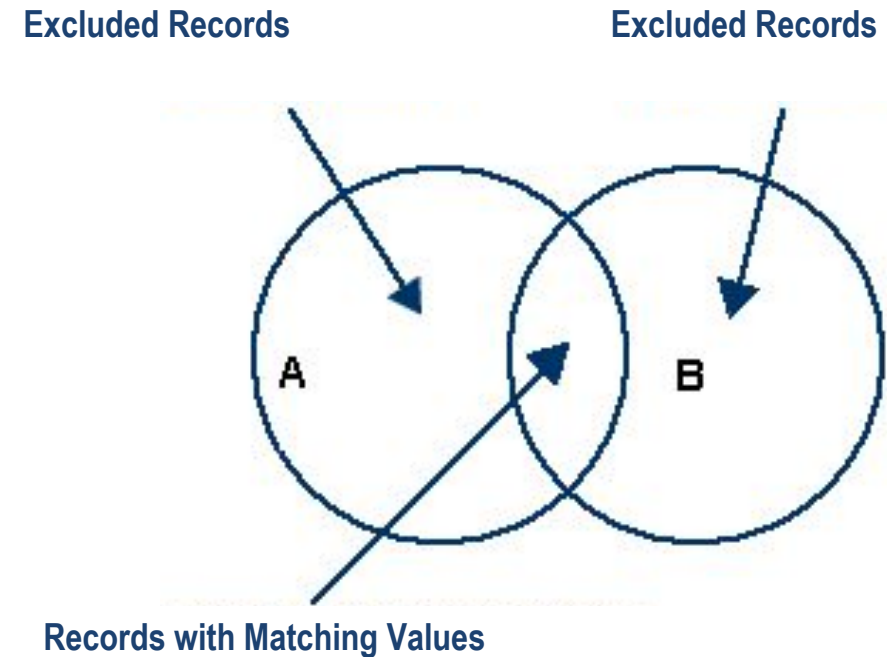
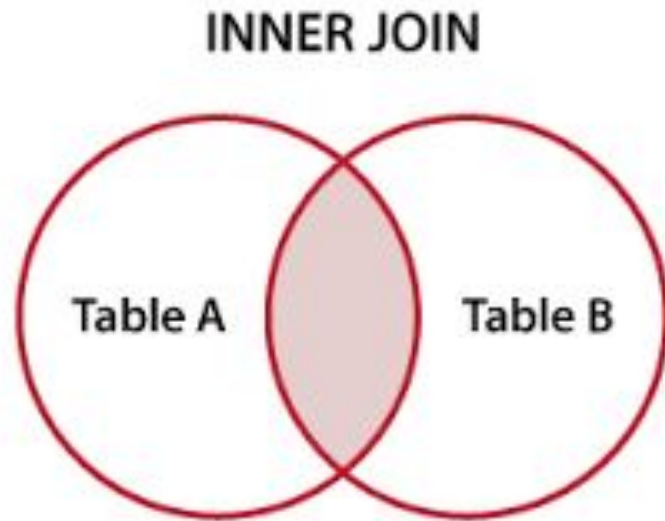
JOINS & UNIONS IN SQL

INTRODUCTION: JOINS

JOINS

SQL **JOIN**s and Excel's **VLOOKUP** command have similar purposes.

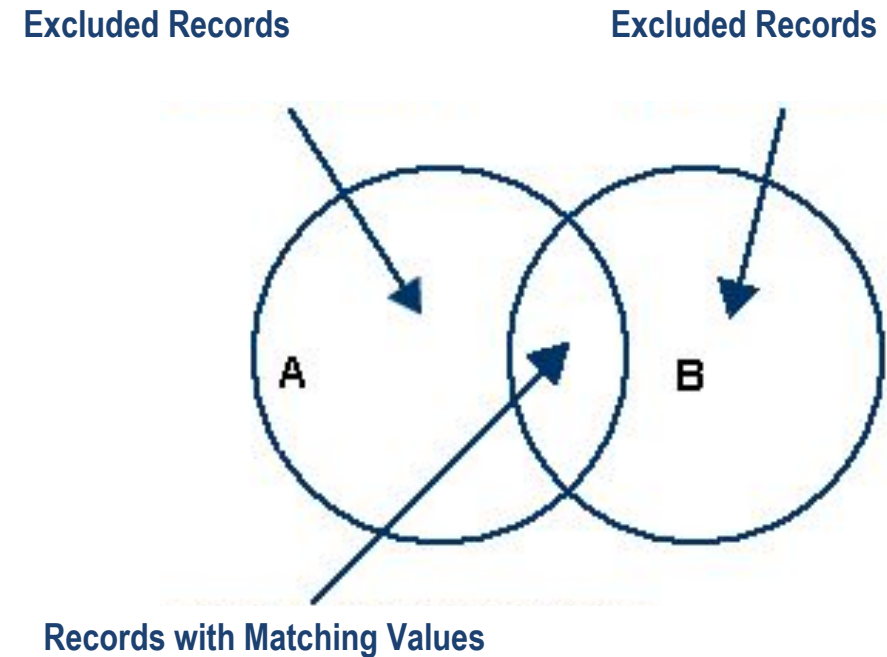
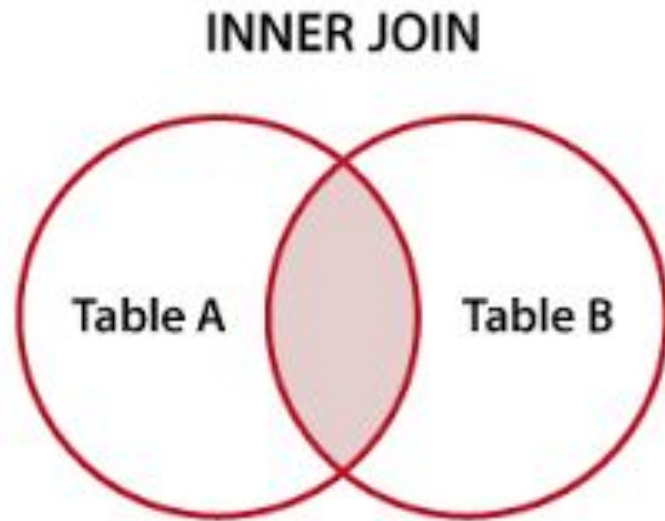
They both connect data sources together in order to use information from both tables to display a desired result.



JOINS

For large relational or object-oriented databases, a **JOIN** enables tables to be connected using common columns, which serve as unique identifiers, or **KEYS**.

We'll start by looking at some inner **JOIN**s.



JOINS & UNIONS IN SQL

DEMO: JOIN SYNTAX

JOINS SYNTAX, EXPLAINED

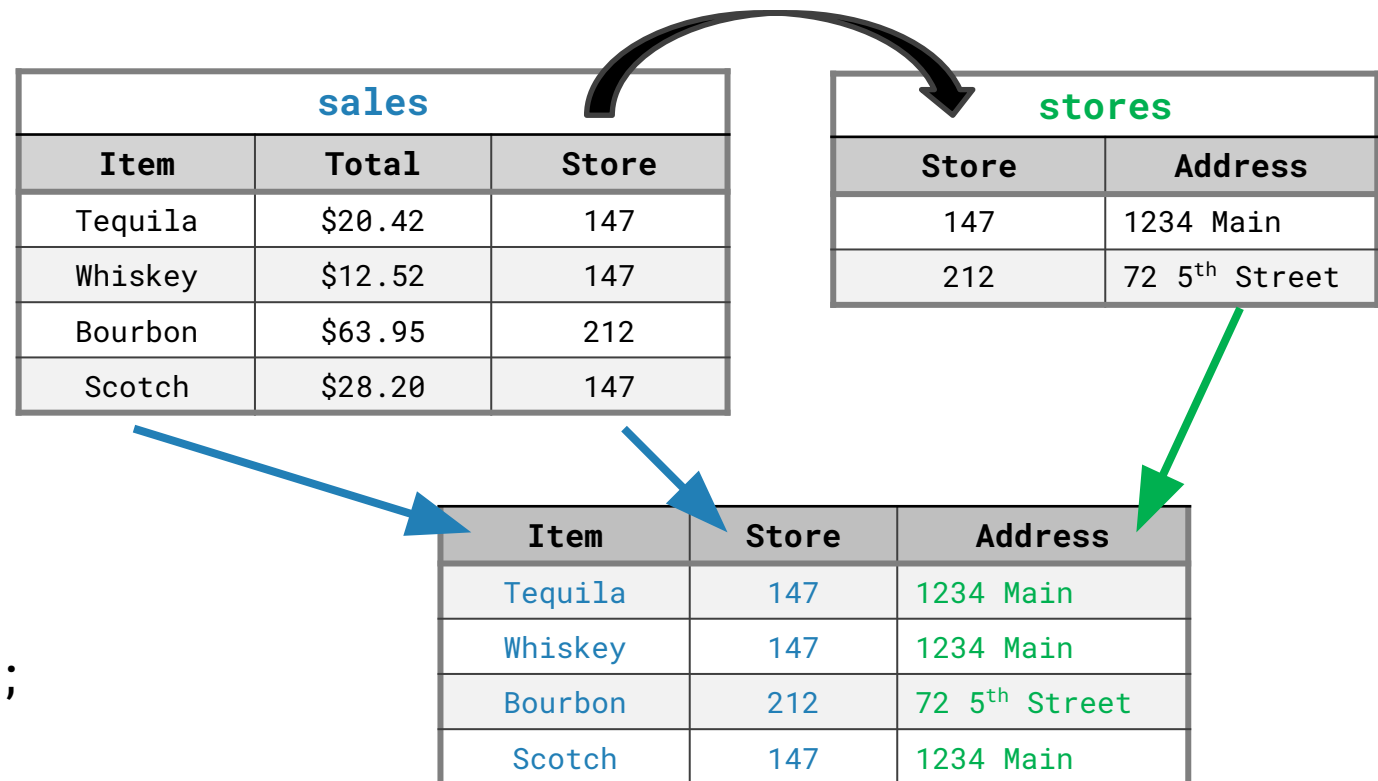
- First** ⇒ Identify which columns of information you are targeting.
- Second** ⇒ Identify the table from which to retrieve information.
- Third** ⇒ Specify the **keys**, or common columns that you wish to join together.

```
SELECT
    sales.item,
    sales.store,
    stores.store_address

FROM
    sales

INNER JOIN stores

    ON sales.store = stores.store;
```



JOINS SYNTAX, EXPLAINED

Each column name must have a prefix that specifies its source. Labeling is accomplished by writing out the source table name **or** by using an **ALIAS**.

ALIASES are designated in the **FROM** statement.

```
SELECT
    stores.store,
    sales.item,
    sales.total
FROM
    sales
INNER JOIN stores
    ON sales.store = stores.store;
```

```
SELECT
    b.store,
    a.item,
    a.total
FROM
    sales a
INNER JOIN stores b
    ON a.store = b.store;
```

JOINS SYNTAX, EXPLAINED

When you create a **JOIN**, each table can have an **ALIAS** and each COLUMN is connected to the TABLE by the **ALIAS**.

An **ALIAS** is a shorthand name given to tables or columns in a table that you intend to reference repeatedly. Here, each column is connected to the table via an **ALIAS**.

table1 **a** → **a**.column1

table1 uses the alias **a**.

table2 **b** → **b**.column4

table2 uses the alias **b**.

JOINS SYNTAX, EXPLAINED

Next, specify the type of **JOIN** you want. An **INNER JOIN** is used by default.

Specify the connection, by column name, on which you want to link tables.

ON **a**.column_name1 = **b**.column_name4 → With alias for source table.

USING(column_name) → Only if the columns have **same** name in each table.

JOINS & UNIONS IN SQL

GUIDED PRACTICE: JOINING SALES TO PRODUCTS

JOINING SALES TO PRODUCTS



EXERCISE

Let's alter this code a bit so it will work better:

```
SELECT b.item_no, b.item_description, a.total  
FROM the Sales table and the Products table.
```

Next, write a query to perform the following:

1. Find the common field on which to **JOIN** them.
2. Limit results to the first 100

JOINING SALES TO PRODUCTS



EXERCISE

Solution:

```
SELECT b.item_no, b.item_description, a.total  
FROM sales a  
INNER JOIN products b  
ON a.item = b.item_no  
LIMIT 100;
```

JOINING SALES TO STORES



EXERCISE

Write a query to discover which **DISTINCT** products were sold in Mason City, IA.

1. Return the product description, category, and store address columns.
2. Qualify **DISTINCT**
3. Select appropriately aliased items **FROM** the Sales and Stores tables.
4. Connect the matching columns with the **USING** structure.

JOINING SALES TO STORES



EXERCISE

Sample Solution: **DISTINCT** products in Mason City, IA

```
SELECT DISTINCT a.description, a.category_name,  
b.store_address  
FROM sales a  
INNER JOIN stores b  
ON a.store = b.store  
WHERE b.store_address LIKE '%Mason City%';
```

JOINS & UNIONS IN SQL

GUIDED PRACTICE: JOINING MULTIPLE TABLES

JOINING MULTIPLE TABLES

Here's an example of joining multiple tables together. Notice that the code repeats itself.

Tip: It can be helpful to sketch and wireframe how your **JOINS** will relate on a piece of paper before coding.

```
SELECT
    b.field1, a.field2, a.field3, c.field4
FROM table1 a
INNER JOIN table2 b
    ON a.field1 = b.field1
INNER JOIN table3 c
    ON a.field1 = c.field1
ORDER BY b.field1
```

JOINS & UNIONS IN SQL

INDEPENDENT PRACTICE: JOINING MULTIPLE TABLES

JOINING MULTIPLE TABLES: PRACTICE NO. 1



EXERCISE

Using Sales as our primary table, create links to all of the other tables in the Iowa Liquor Sales Database. The result should be one query with several **JOINS**.

Before going into SQL, practice wireframing your **JOINS** on a piece of paper.

Your query should:

1. Include county from the County table, store from the Stores table, store name from the Stores table, item name, case_cost from the Products table, and total from the Sales table.
2. Limit results to 1,000.

JOINING MULTIPLE TABLES: PRACTICE NO. 1

Solution query:

```
SELECT d.county, a.store, b.name, a.item,  
       c.case_cost, a.total  
FROM sales a  
INNER JOIN stores b  
ON a.store=b.store  
  
INNER JOIN products c  
ON a.item=c.item_no  
  
INNER JOIN counties d  
ON a.county = d.county  
LIMIT 1000;
```



EXERCISE

JOINING MULTIPLE TABLES: PRACTICE NO. 2

Using Sales as our primary table, create links to the Products and Stores tables within the Iowa Liquor Sales Database (several **INNER JOINS**).

Before proceeding, plan your **JOINS** out in a wireframe.



EXERCISE

Your query should:

1. List the store number (from Sales), category_name (from Sales), and two aggregated columns: average bottle_price (from Products) and average total price (from Sales).
2. Use a compounded **WHERE** clause to limit the calculations to the sales of tequila (category_name from Sales) from active stores in Mason City, Iowa.
3. Group and sort the data by the store number.

JOINING MULTIPLE TABLES: PRACTICE NO. 2

Desired data output:



Data Output					Explain	Messages	History
	store integer	category_name text	avg_cost numeric	avg_sales_price numeric			
1	2515	TEQUILA	13.20	164.51			
2	2582	TEQUILA	13.21	140.72			
3	2652	TEQUILA	13.19	146.68			
4	3528	TEQUILA	13.75	214.64			
5	3713	TEQUILA	12.96	189.00			
6	4104	TEQUILA	10.02	169.38			
7	4367	TEQUILA	9.82	53.85			
8	4372	TEQUILA	12.25	220.44			
9	4376	TEQUILA	11.12	71.31			
10	4615	TEQUILA	13.85	184.77			
11	4755	TEQUILA	10.49	188.76			
12	4955	TEQUILA	12.07	77.03			
13	5034	TEQUILA	13.28	73.84			

JOINING MULTIPLE TABLES: PRACTICE NO. 2

Solution query without aliases:

```
SELECT sales.store, sales.category_name,  
       ROUND(AVG(CAST(products.bottle_price AS DEC)),2) AS avg_cost,  
       ROUND(AVG(sales.total),2) AS avg_sales_price  
FROM sales  
INNER JOIN products  
    ON sales.item = products.item_no  
INNER JOIN stores  
    ON stores.store = sales.store  
WHERE sales.category_name LIKE '%TEQUILA%' AND  
       stores.store_address LIKE '%Mason City%' AND  
       store_status='A'  
GROUP BY sales.store, sales.category_name  
ORDER BY 1;
```



EXERCISE

JOINS & UNIONS IN SQL

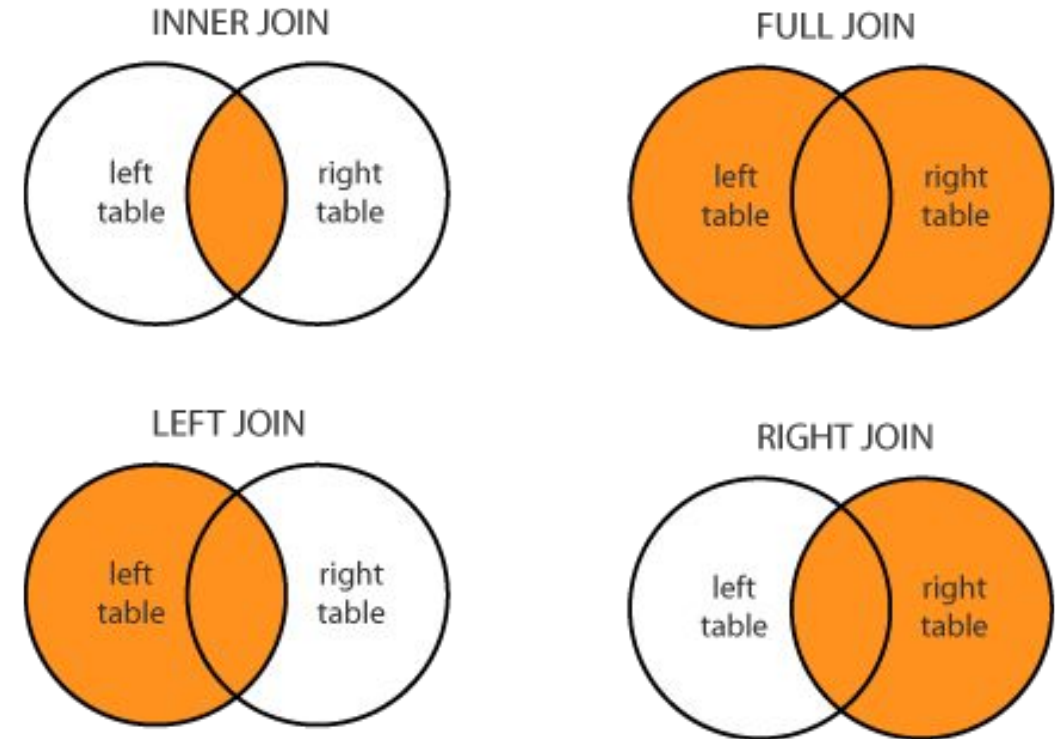
CONCLUSION

RECAP UNIONS AND JOINS

UNIONS and **JOINS** are just a few ways to connect multiple tables.

Next, we'll learn more about different types of **JOINS**, where you can prioritize one table and specify selected parts of another table for your output.

We'll cover this tomorrow!



WHAT ELSE HAVE WE COVERED?

In today's lesson, we learned how to:

1. Explain some SQL methods for appending similar data together.
2. Explore ways to combine data from different tables together.
3. Use the SQL commands **JOIN** and **UNION** to answer data questions.
4. Apply **JOINS** to connect data from multiple sources.

JOINS & UNIONS IN SQL



ACTIVITY

DIRECTIONS

1. Pair up and take 10 minutes to discuss:
 - Key takeaways from today.
 - What you can do to make these takeaways actionable and valuable for your teams.
 - Thought starters: Better knowledge of keys in databases, aligning with database teams on data organization.

DELIVERABLE

Be prepared to share your answers with the class.

JOINS & UNIONS IN SQL

Q&A

JOINS & UNIONS IN SQL

RESOURCES

JOINS & UNIONS IN SQL

RESOURCES

- Microsoft reference material on UNIONS:
<https://docs.microsoft.com/en-us/sql/t-sql/language-elements/set-operators-union-transact-sql>
- INNER JOIN tutorial: <http://www.sqltutorial.org/sql-inner-join/>
- “What is the Difference Between a JOIN and a UNION?”:
<https://www.essentialsql.com/what-is-the-difference-between-a-join-and-a-union/>
- “What is the difference between a primary and unique key?”
<https://www.essentialsql.com/primary-and-unique-key/>

JOINS & UNIONS IN SQL

CREDITS

UNION and JOIN graphical illustration from EssentialSQL.com:
<https://goo.gl/FQXykj>.