

What is SQL

Setting up the Environment

Files to Download

Installation

Dataset

Structure of an SQL environment

Servers

Databases

Schemas

Tables

Different Database Types

SQL Data Types

Basic Querying

Obtaining Data

SELECT

LIMIT

DISTINCT

WHERE

ORDER BY

Aliasing

Comments

Aggregations

GROUP BY

AVG

COUNT

COUNT (DISTINCT)

SUM

MIN

MAX

HAVING

Combining Data

Joins and Unions

UNION

JOIN (LEFT, INNER, OUTER, RIGHT)

Advanced Querying

Subquerying

Advanced Filtering

LIKE

Wildcard Operators

BETWEEN

IN

Advanced Aggregation

Window Functions

OVER and PARTITION BY

ROW\_NUMBER()

RANK()

DENSE\_RANK()

NTILE

LAG

LEAD

Manipulating Tables

Basics

CREATE

DROP

ALTER

Views

Tips

Further Research

References

# What is SQL

SQL (pronounced S.Q.L. or Sequel, no one really cares) stands for Structured Querying Language and is a language that is used to query, form, and manipulate databases.

## Setting up the Environment

### Files to Download

SQL Course Public Downloads Folder - Google Drive

 <https://drive.google.com/drive/folders/1t1l7mcnhxN0rvYH-W...>

3

### Installation

For this tutorial we're going to be using SQLite which is a database engine that works in a single application. In order to access it we'll be using DBeaver. DBeaver is an open-source database manager that connects to almost all database engines.

**NOTE: Use DBeaver Community, it's free**

DBeaver

Free multi-platform database tool for developers, database administrators, analysts and all people who need to work with databases. Supports all popular databases: MySQL, PostgreSQL, SQLite, Oracle, DB2, SQL Server,

 <https://dbeaver.io/>

### Dataset

The dataset we'll be using is from the University of California Irvine's Machine Learning Repository

[archive.ics.uci.edu](http://archive.ics.uci.edu)

Abstract: This data studies whether a person will accept the coupon recommended to him in different driving scenarios  
Source: Tong Wang, tong-wang '@' uiowa.edu, University of Iowa Cynthia Rudin, cynthia '@'

<https://archive.ics.uci.edu/ml/datasets/in-vehicle+coupon+recommendation>

# Structure of an SQL environment

## Servers

The computer that contains your database or databases. This is typically your highest level of "container" for databases.

## Databases

A database is a collection of Schemas and Tables that we can interact with.

## Schemas

Schemas are the highest level of organization in many Databases and can contain multiple tables.

## Tables

Tables can be thought of in the same way you think about sheets in Excel. It's a collection of columns and rows that we'll be interacting with.

## Different Database Types

I'll be talking about SQL like it's a standardized language and although it technically is, in practice SQL databases you'll most likely be interacting with are going to be created and run by major corporations who will sprinkle their own flavor of SQL on top of or in lieu of standard SQL. The commands we'll be focusing in this course are pretty standard and shouldn't be drastically different from that which you'll be using in most other databases. Some of the biggest databases are listed below:

MySQL - Open Source - Current Development by Oracle

SQL Server - Created by Microsoft

PostgreSQL - Open Source

Oracle Database - Created by Oracle

## SQL Data Types

Depending on the database type that you use, the datatypes that you'll be working with will be different. Here are the basic types and what they might be referred to by in your database.

**Text types:** Used to store text, can even store numbers as text

- CHAR 💬 2
- VARCHAR 💬 2
- TEXT 💬 1

**Numerical Types:** Used to store numbers

- INTEGER
- FLOAT
- REAL
- NUMERIC
- BOOL
- BOOLEAN

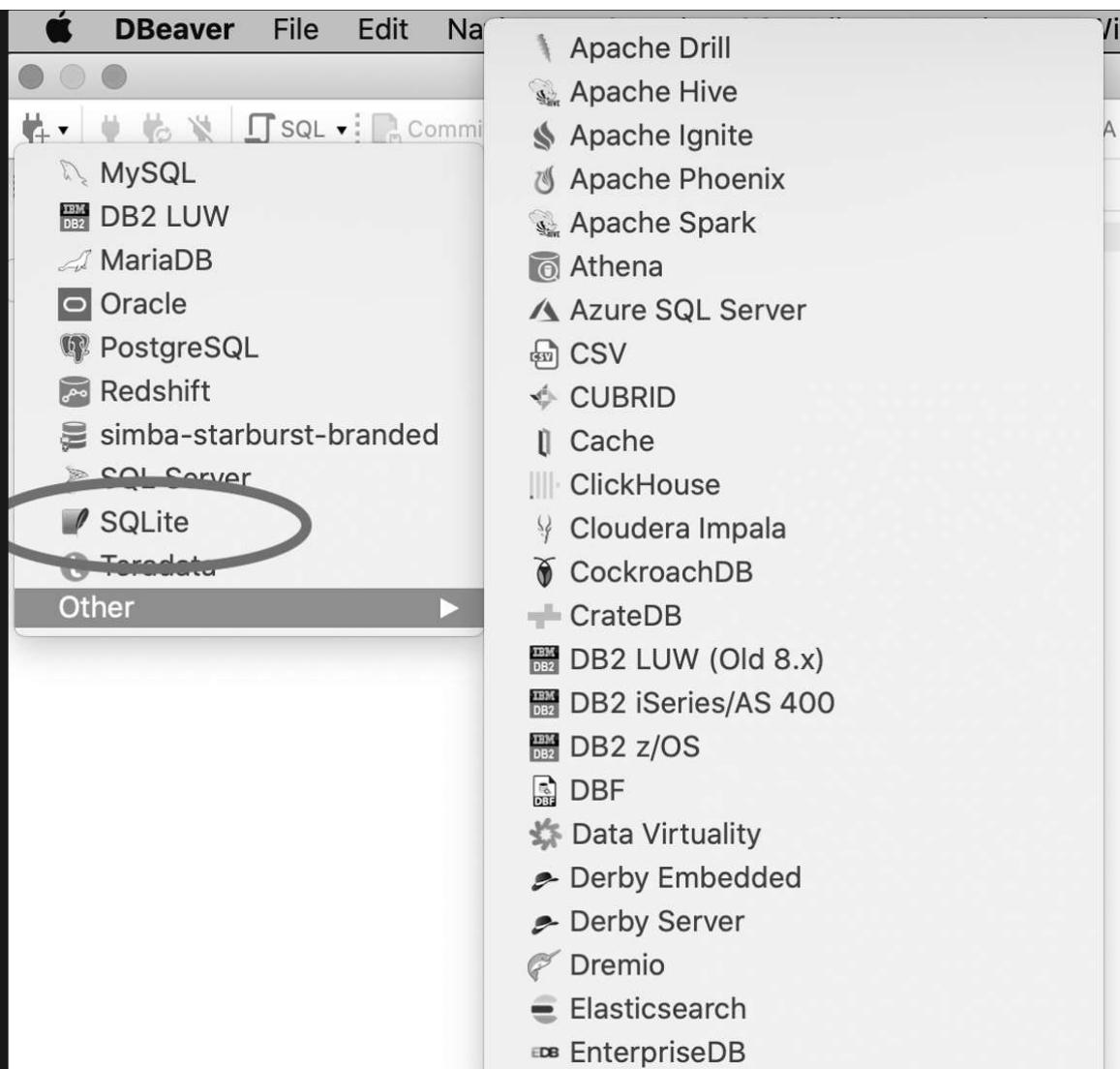
**Date Types:** Used to store dates and times

**Binary Types:** Large files stores using "1's and 0's", hence binary

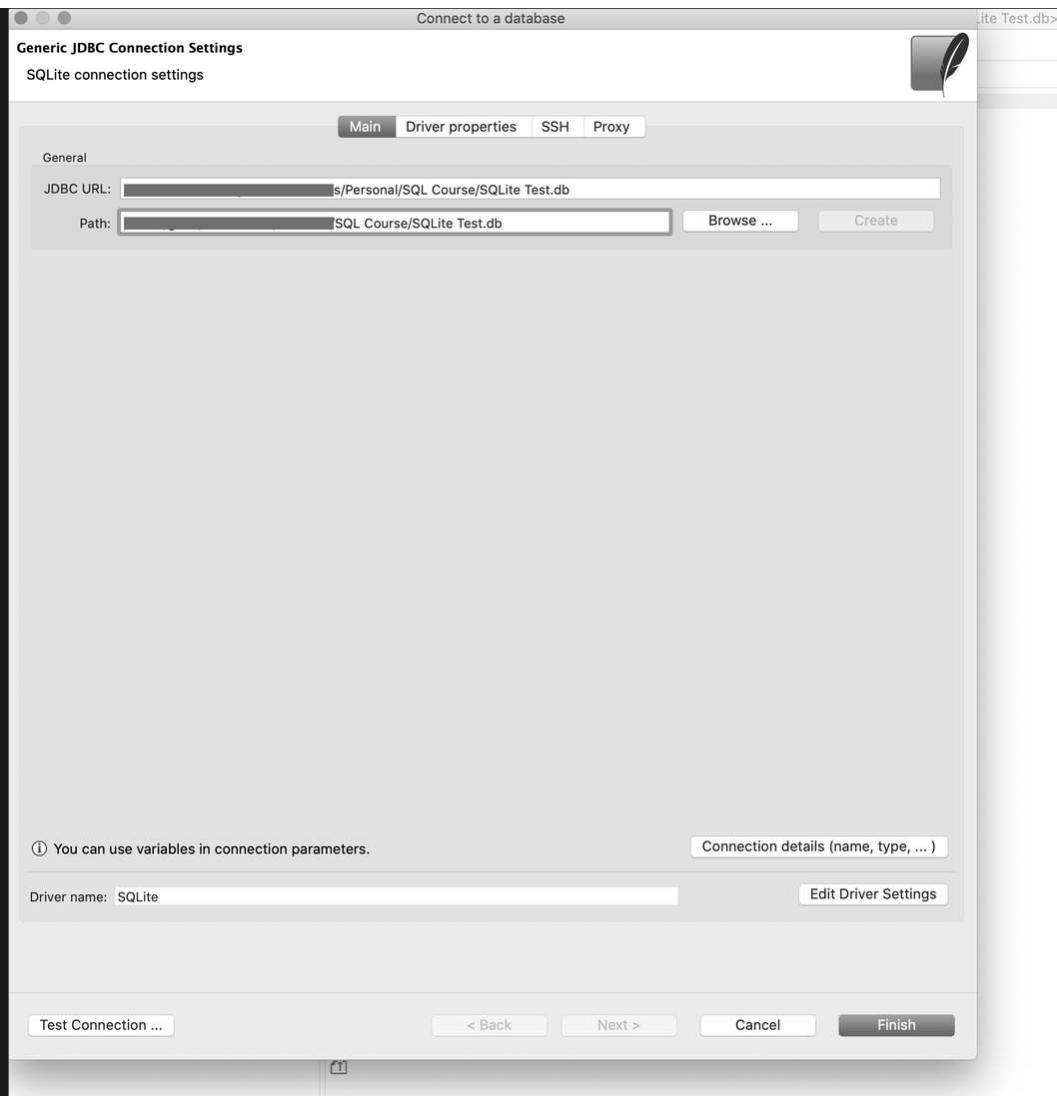
- TINYBLOB
- BLOB

## Basic Querying

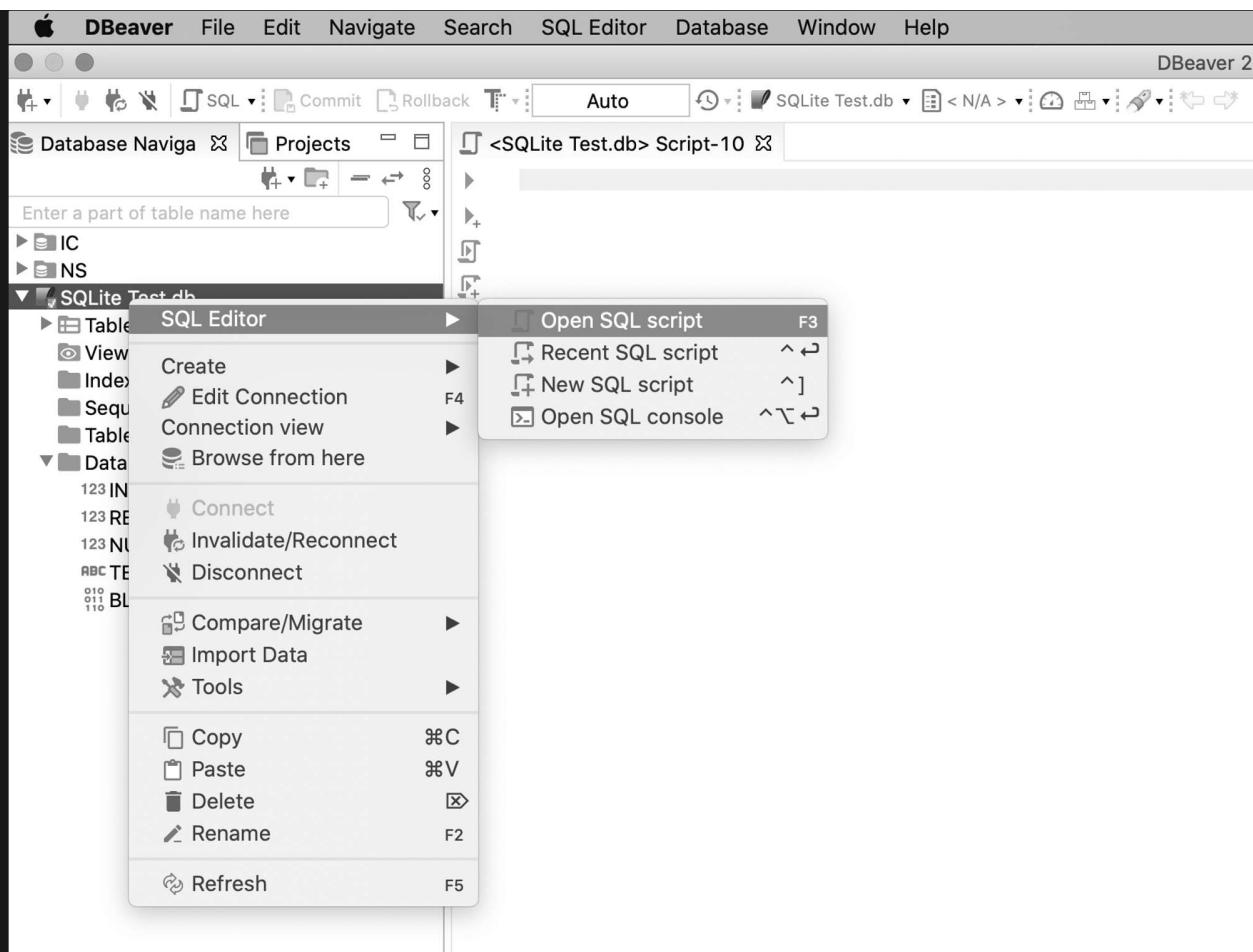
After opening DBeaver and downloading the database file from the link mentioned at the top of the page, create a new connection and select the option for SQLite.



Open up the database file you downloaded



If you're using DBeaver, then right click our database and Open a SQL script



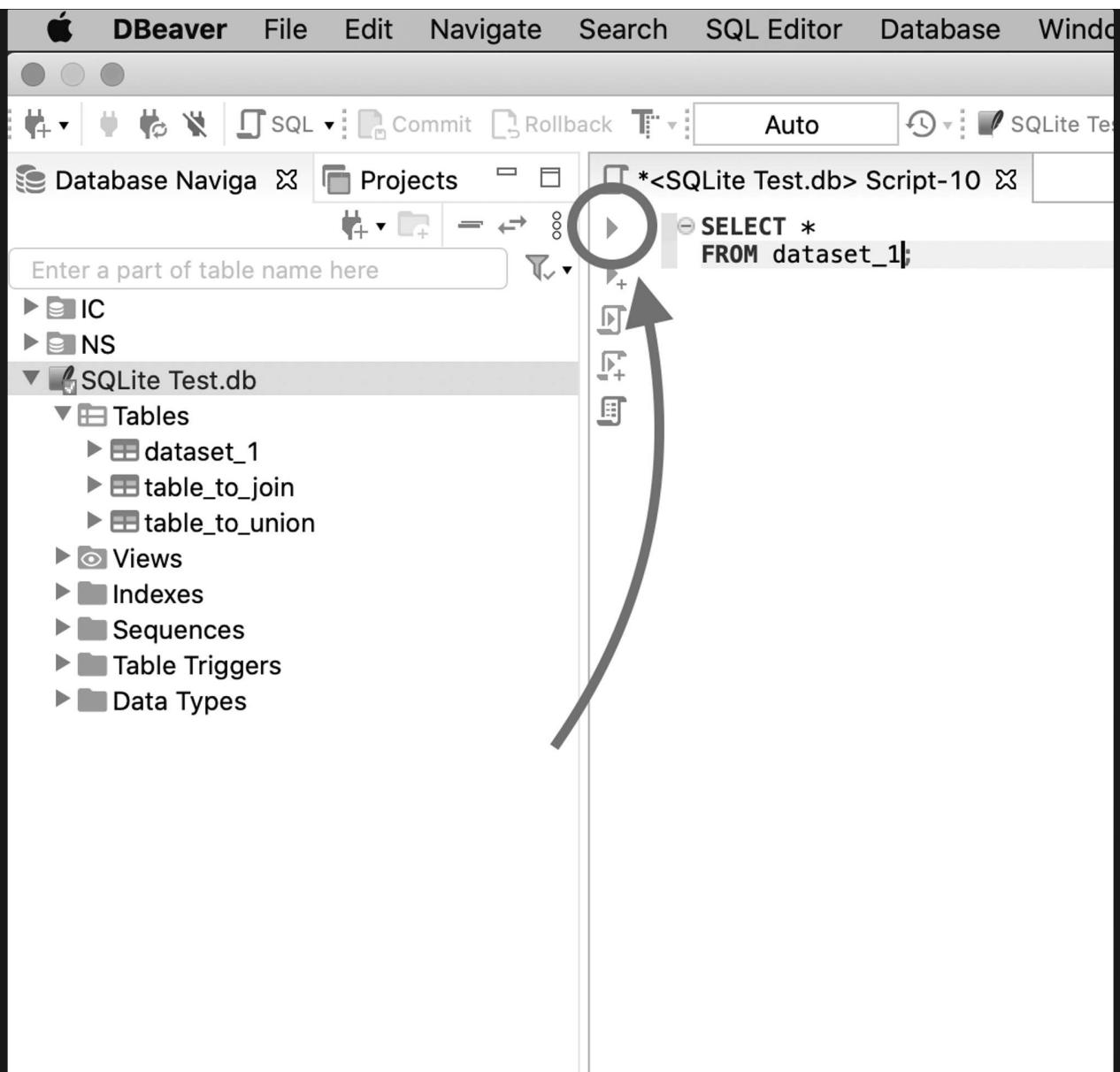
## Obtaining Data

### SELECT

The most basic query, this is used to pull data from a database.

```
SELECT * FROM dataset_1;
```

You can run this query by pasting the text into the SQL Editor and clicking on this play button



In the above statement you have a couple of parts:

`SELECT *` this is the command

`FROM` this tells the Database where to pull data from

`dataset_1` this is the table you're pulling data from

`;` the semicolon at the end is how you end an SQL statement. Some databases require it, others don't so I recommend always using one.

The above command selects all columns in the dataset, you can also specify columns like so:

```
SELECT weather, temperature FROM dataset_1;
```

This will only select the `weather` and `temperature` columns from the dataset.

## LIMIT

Can be used to limit the amount of data that is pulled into a query, very useful when first exploring the data.

**Query:**

```
select * FROM dataset_1 LIMIT 10;
```

**Result:**

	RBC destination	RBC passanger	RBC weather	123 temperature	RBC time	RBC coupon
1	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House
3	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House
5	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House
6	No Urgent Place	Friend(s)	Sunny	80	6PM	Restaurant(<20)
7	No Urgent Place	Friend(s)	Sunny	55	2PM	Carry out & Take away
8	No Urgent Place	Kid(s)	Sunny	80	10AM	Restaurant(<20)
9	No Urgent Place	Kid(s)	Sunny	80	10AM	Carry out & Take away
10	No Urgent Place	Kid(s)	Sunny	80	10AM	Bar

## DISTINCT

Gets the unique values from a column

**Query:**

```
SELECT DISTINCT passanger FROM dataset_1;
```

**Result:**

	ABC passanger	↑↓
1	Alone	
2	Friend(s)	
3	Kid(s)	
4	Partner	

## WHERE

Query:

```
SELECT * FROM dataset_1 WHERE destination = 'Home';
```

Result:

	ABC destination	↑↓	ABC passanger	↑↓	ABC weather	↑↓	123 temperature	↑↓	RBC time	↑↓	RBC coupon	↑↓
1	Home		Alone		Sunny		55		6PM		Bar	
2	Home		Alone		Sunny		55		6PM		Restaurant(20-50)	
3	Home		Alone		Sunny		80		6PM		Coffee House	
4	Home		Alone		Sunny		55		6PM		Bar	
5	Home		Alone		Sunny		55		6PM		Restaurant(20-50)	
6	Home		Alone		Sunny		80		6PM		Coffee House	
7	Home		Alone		Sunny		55		6PM		Bar	
8	Home		Alone		Sunny		55		6PM		Restaurant(20-50)	
9	Home		Alone		Sunny		80		6PM		Coffee House	
10	Home		Alone		Sunny		55		6PM		Bar	
11	Home		Alone		Sunny		55		6PM		Restaurant(20-50)	
12	Home		Alone		Sunny		80		6PM		Coffee House	

## ORDER BY

Query:

```
SELECT * FROM dataset_1 ORDER BY coupon;
```

Result:

	RBC destination	RBC passanger	RBC weather	123 temperature	RBC time	RBC coupon
1	No Urgent Place	Kid(s)	Sunny	80	10AM	Bar
2	Home	Alone	Sunny	55	6PM	Bar
3	Work	Alone	Sunny	55	7AM	Bar
4	No Urgent Place	Friend(s)	Sunny	80	10AM	Bar
5	Home	Alone	Sunny	55	6PM	Bar
6	Work	Alone	Sunny	55	7AM	Bar
7	No Urgent Place	Friend(s)	Sunny	80	10AM	Bar
8	Home	Alone	Sunny	55	6PM	Bar
9	Work	Alone	Sunny	55	7AM	Bar
10	No Urgent Place	Kid(s)	Sunny	80	10AM	Bar
11	Home	Alone	Sunny	55	6PM	Bar
12	Work	Alone	Sunny	55	7AM	Bar
13	No Urgent Place	Friend(s)	Sunny	80	10AM	Bar
14	Home	Alone	Sunny	55	6PM	Bar
15	Work	Alone	Sunny	55	7AM	Bar
16	No Urgent Place	Friend(s)	Sunny	80	10AM	Bar
17	Home	Alone	Sunny	55	6PM	Bar
18	Work	Alone	Sunny	55	7AM	Bar
19	No Urgent Place	Kid(s)	Sunny	80	10AM	Bar

## Aliasing

Query:

```
SELECT destination as 'Destination' FROM dataset_1;
```

Result:

	RBC Destination
1	No Urgent Place
2	No Urgent Place
3	No Urgent Place
4	No Urgent Place
5	No Urgent Place
6	No Urgent Place
7	No Urgent Place
8	No Urgent Place
9	No Urgent Place
10	No Urgent Place
11	No Urgent Place
12	No Urgent Place
13	No Urgent Place
14	Home

## Comments

Query:

```
SELECT * -- This is a comment FROM dataset_1;
```

Result:

	RBC Destination
1	No Urgent Place
2	No Urgent Place
3	No Urgent Place
4	No Urgent Place
5	No Urgent Place
6	No Urgent Place
7	No Urgent Place
8	No Urgent Place
9	No Urgent Place
10	No Urgent Place

## Aggregations

### GROUP BY

Query:

```
SELECT occupation FROM dataset_1 GROUP BY occupation;
```

Result:

	RBC occupation	▼
1	Architecture & Engineering	
2	Arts Design Entertainment Sports & Media	
3	Building & Grounds Cleaning & Maintenance	
4	Business & Financial	
5	Community & Social Services	
6	Computer & Mathematical	
7	Construction & Extraction	
8	Education&Training&Library	
9	Farming Fishing & Forestry	
10	Food Preparation & Serving Related	
11	Healthcare Practitioners & Technical	
12	Healthcare Support	
13	Installation Maintenance & Repair	
14	Legal	
15	Life Physical Social Science	
16	Management	
17	Office & Administrative Support	
18	Personal Care & Service	
19	Production Occupations	
20	Protective Service	
21	Retired	
22	Sales & Related	
23	Student	
24	Transportation & Material Moving	
25	Unemployed	

## AVG

Query:

```
SELECT weather, AVG(temperature) AS 'avg_temp' FROM dataset_1 GROUP BY weather;
```

Result:

	RBC weather	▼	avg_temp	▼
1	Rainy		55	
2	Snowy		30	
3	Sunny		68.9462707319	

## COUNT

Query:

```
SELECT weather, COUNT(temperature) AS 'count_temp' FROM dataset_1 GROUP BY weather;
```

**Result:**

	ABC weather	123 count_temp
1	Rainy	1,210
2	Snowy	1,405
3	Sunny	10,069

## COUNT (DISTINCT)

**Query:**

```
SELECT weather, COUNT(DISTINCT temperature) AS 'count_distinct_temp' FROM dataset_1 GROUP BY weather;
```

**Result:**

	ABC weather	123 count_distinct_temp
1	Rainy	1
2	Snowy	1
3	Sunny	3

## SUM

**Query:**

```
SELECT weather, SUM(temperature) AS 'sum_temp' FROM dataset_1 GROUP BY weather;
```

**Result:**

	ABC weather	123 sum_temp
1	Rainy	66,550
2	Snowy	42,150
3	Sunny	694,220

## MIN

Query:

```
SELECT weather, MIN(temperature) AS 'min_temp' FROM dataset_1 GROUP BY weather;
```

Result:

	ABC weather ↑↓	123 min_temp ↑↓
1	Rainy	55
2	Snowy	30
3	Sunny	30

## MAX

Query:

```
SELECT weather, MAX(temperature) AS 'max_temp' FROM dataset_1 GROUP BY weather;
```

Result:

	ABC weather ↑↓	123 max_temp ↑↓
1	Rainy	55
2	Snowy	30
3	Sunny	80

## HAVING

Query:

```
SELECT occupation FROM dataset_1 GROUP BY occupation HAVING occupation = 'Student';
```

Result:

	RBC occupation	▼ ▲
1	Student	

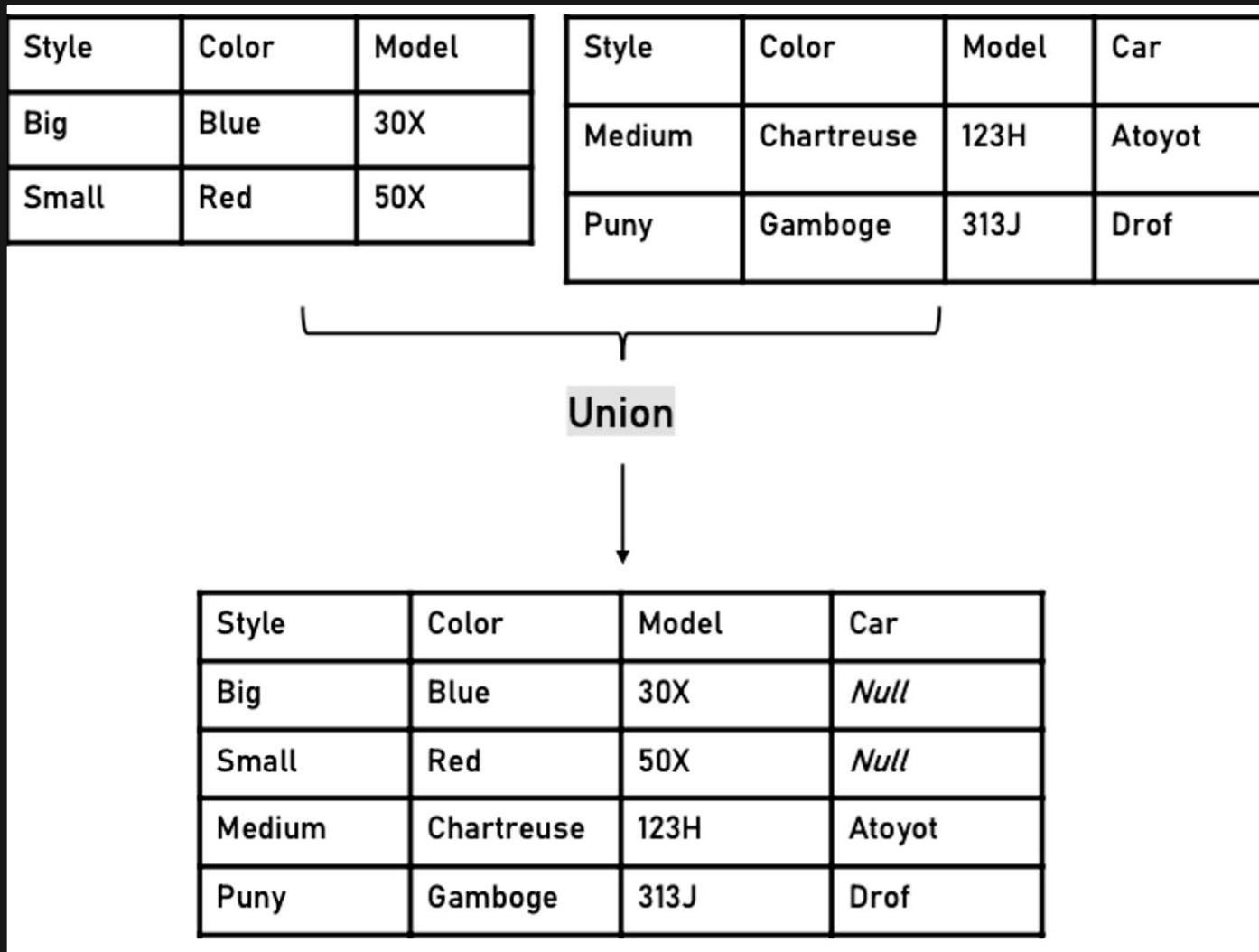
## Combining Data

### Joins and Unions

Oftentimes we'll want to enrich our data by adding more data from other datasets to it. There are two major ways we can combine our datasets together: (The following visualizations will be coming from my FREE Tableau Course)

### Unions

Put simply, a union (SQL Union) is the process of stacking two tables on top of one another. You will usually do this when your data is split up into multiple sections like an excel spreadsheet of a year's sales split by month.



## Joins

Joins combine two tables horizontally. For a join, like a Union you have to have at least two tables, what we call our Left Table and our Right Table. You (mostly) have to have at least one matching column between the two tables, and you will match rows from these columns. The most common way to visualize the types of Joins are through Venn Diagrams.

ID	Color	Model
1	Blue	30X
3	Red	50X

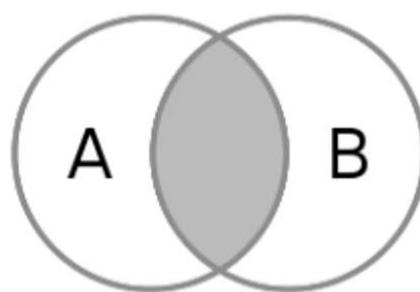
Left Table

ID	Car
1	Atoyot
2	Drof

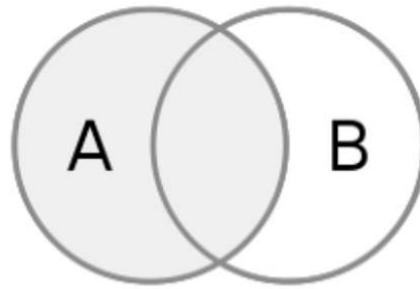
Right Table

**There are four basic joins that you can use.**

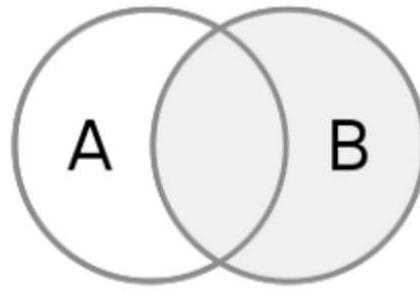
**Inner Join**



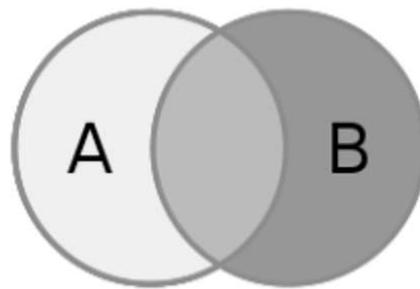
**Left Join**



**Right Join**

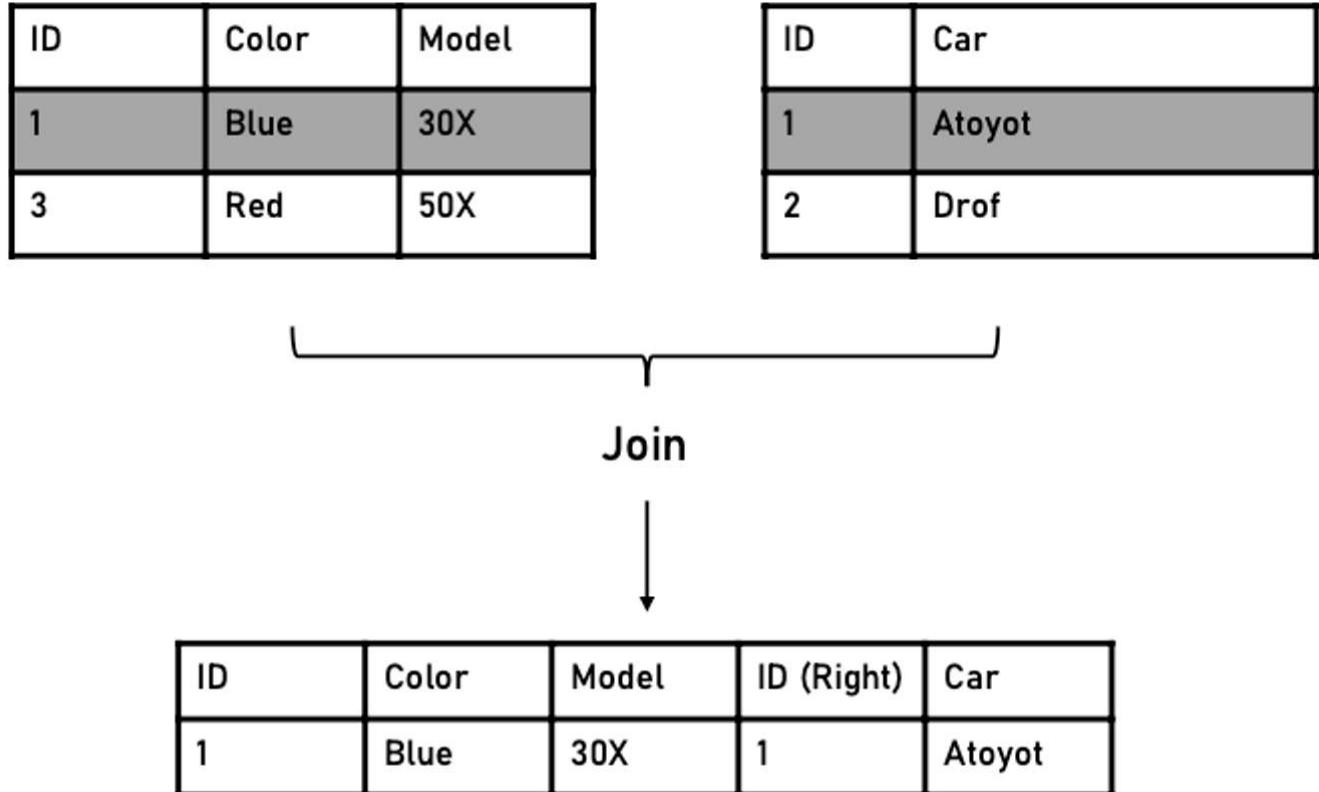


**Full Join**



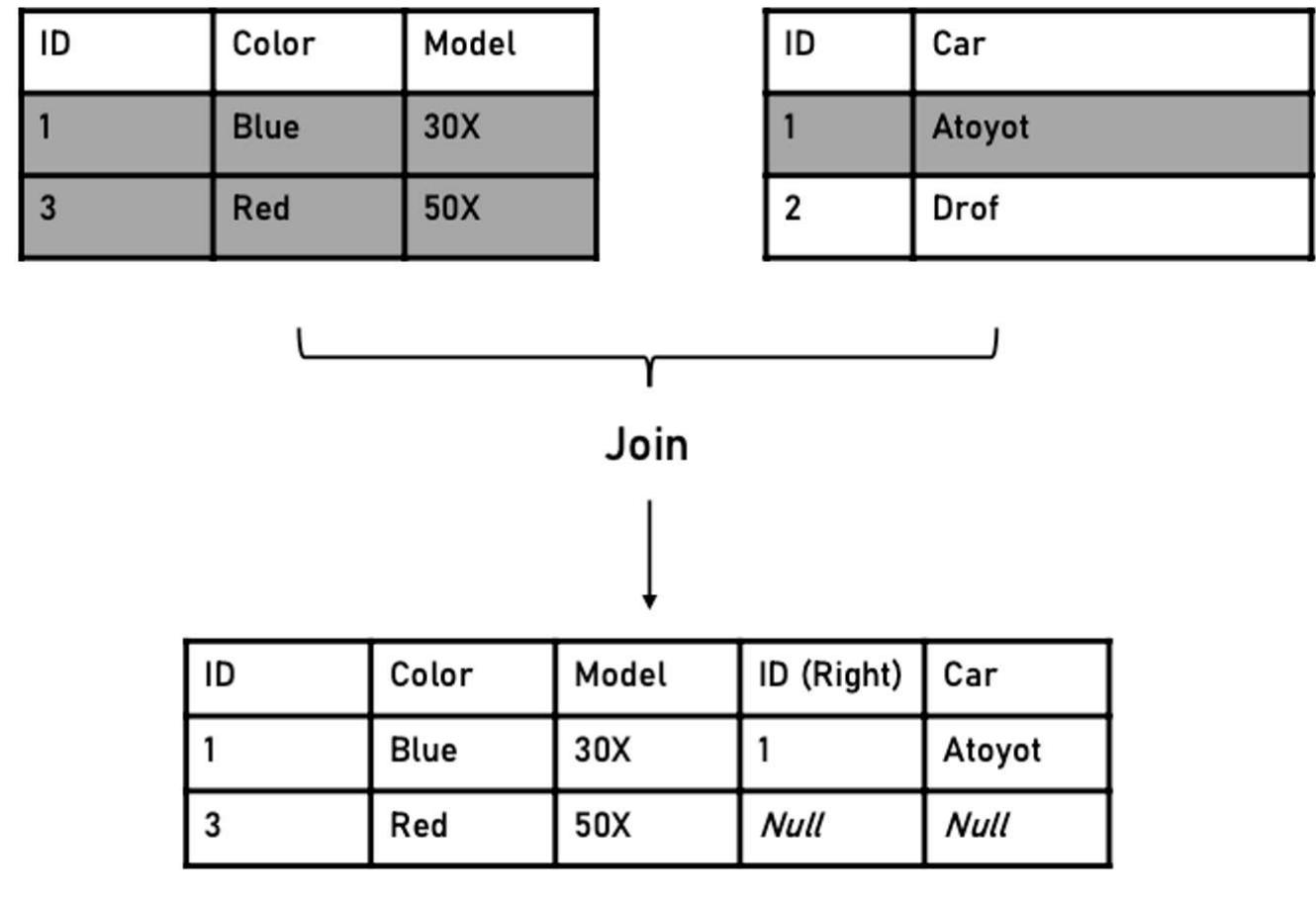
You'll mostly be sticking to Left and Inner Joins. It's worth your time to learn more about Joins because they are some of the most powerful tools you can use to manipulate data. I use Joins basically every single day in my work. For this course we're going to stick with relatively simple Joins.

We're now going to do something called an Inner Join on the [ID] column which will only output exact matches from the [ID] column in our output.



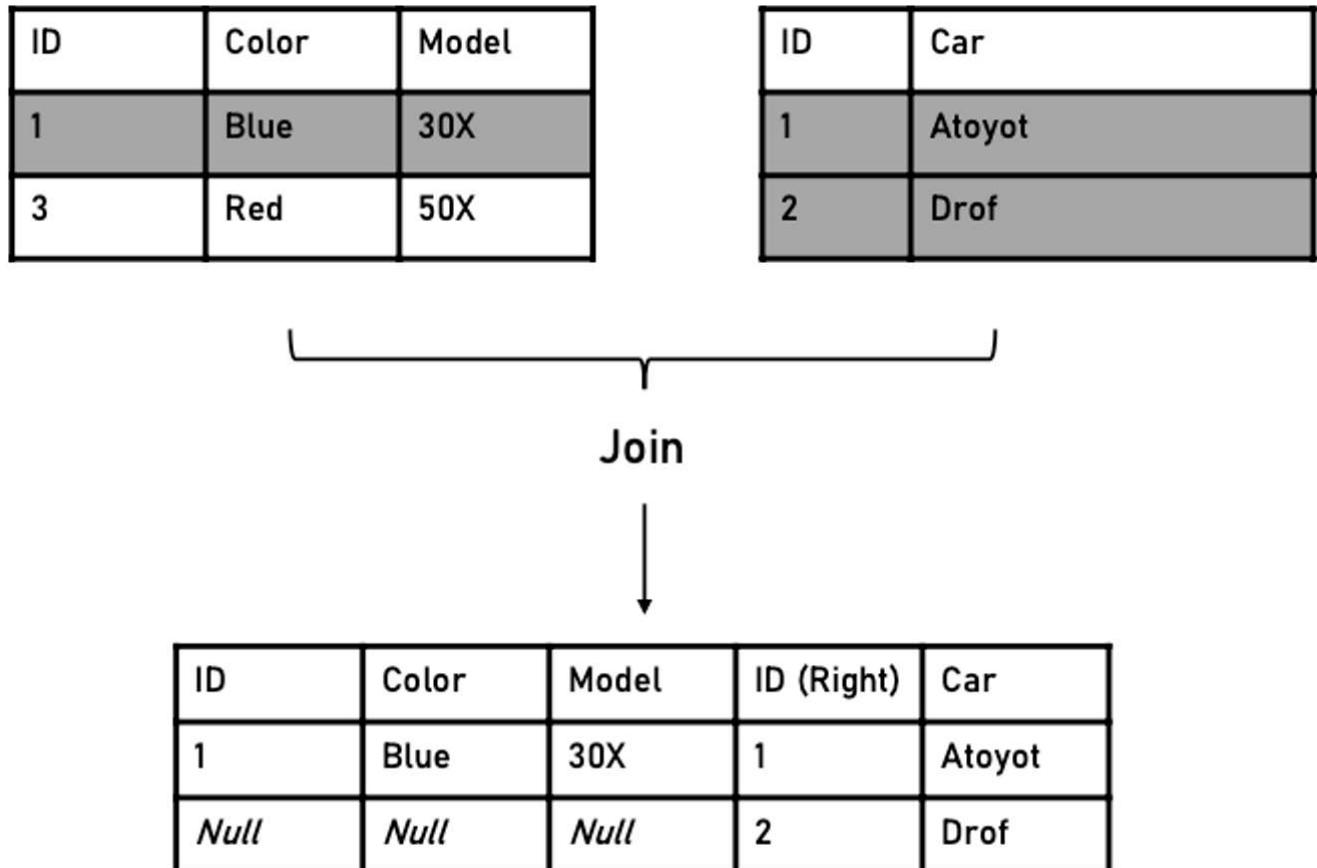
Inner Join

A Left Join keeps all of the data from your Left table and whatever matches from the Right table.



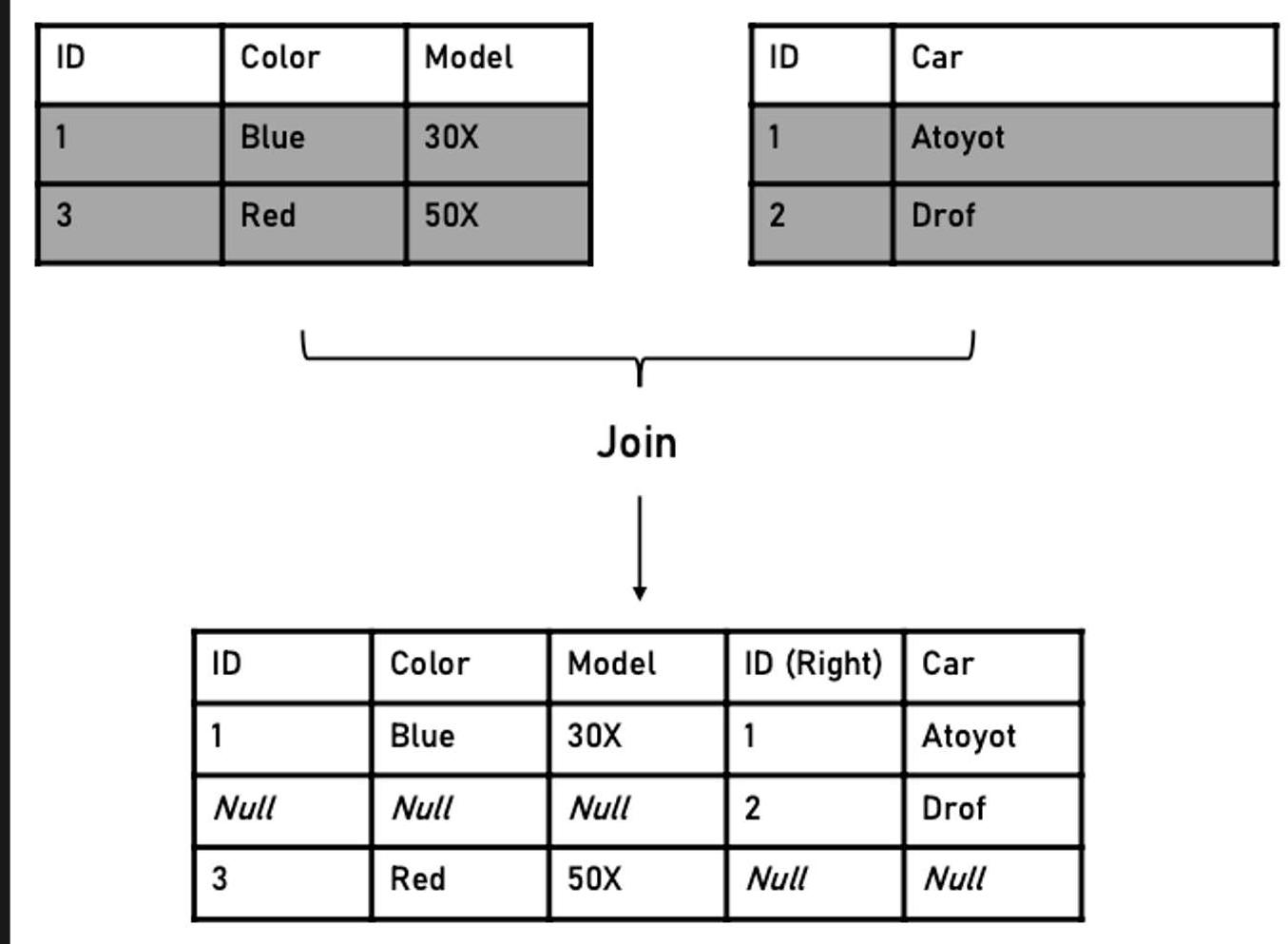
Left Join

A Right Join does the exact opposite and keeps everything from your Right table while only bringing in the matches from the Left table.



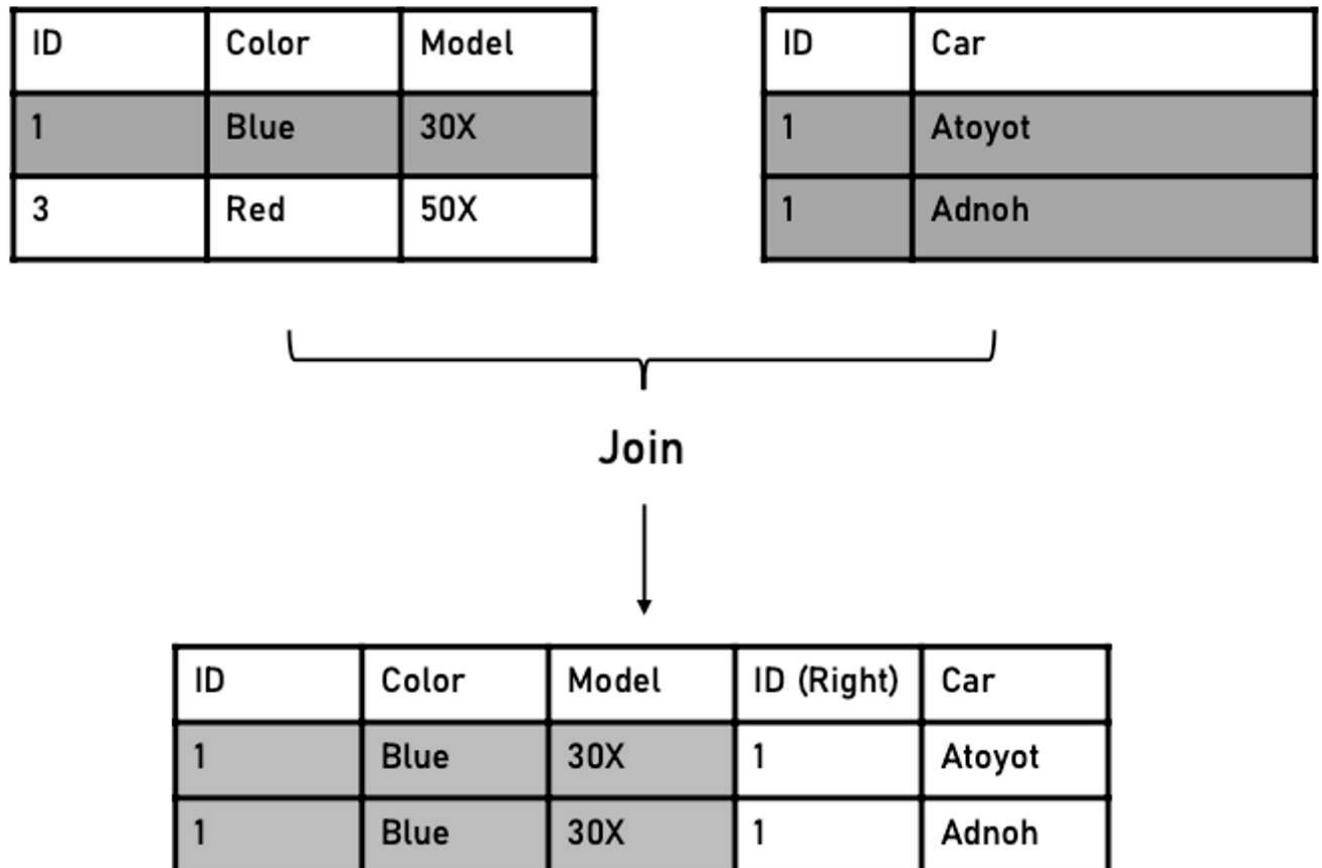
Right Join

A Full Join brings in everything from both tables and matches whatever will match from the columns you specify.



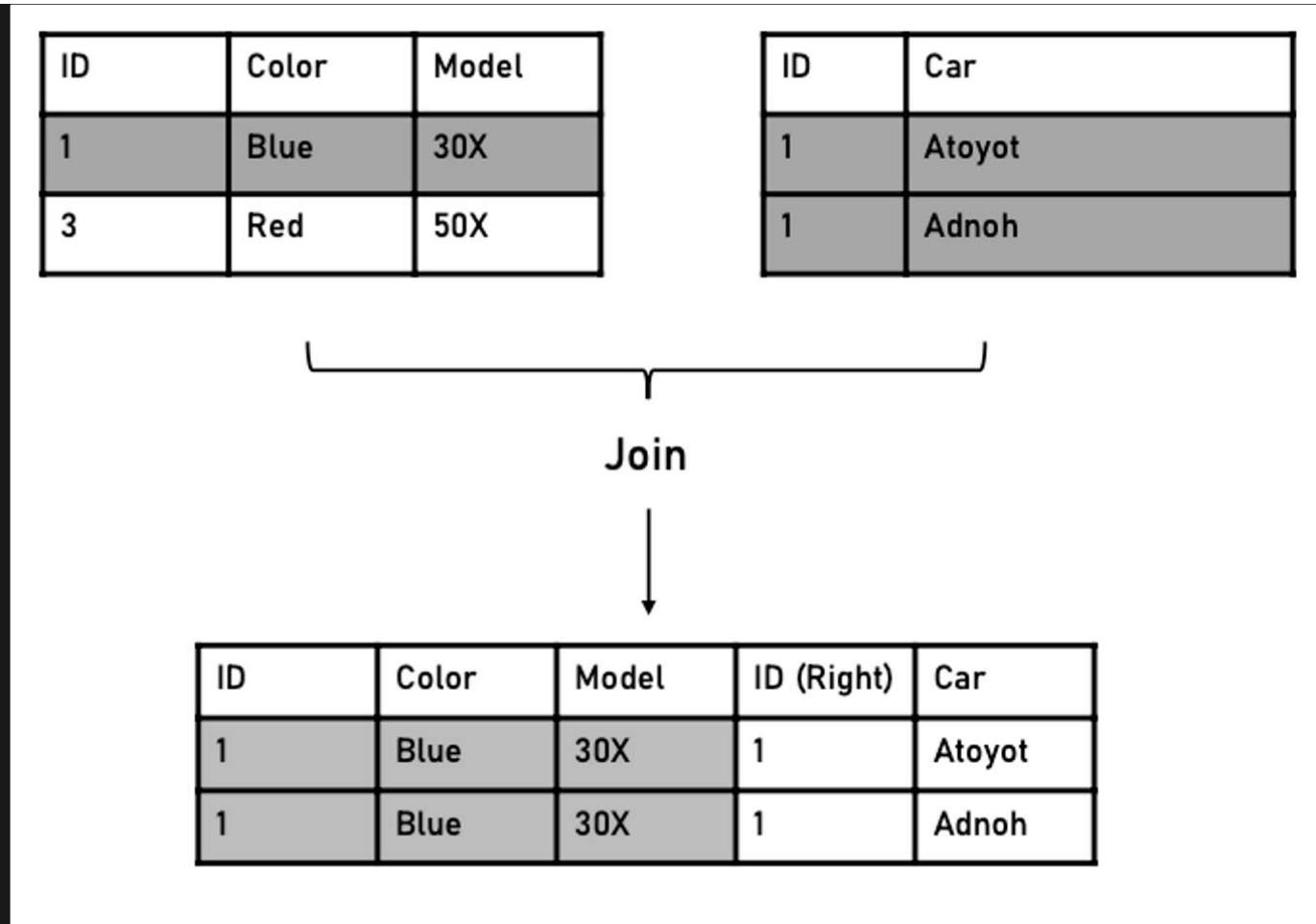
### Common Join Gotchas

Joins can get a bit tricky because of the potential for gotchas when joining two tables. The most common one is row duplication where you accidentally duplicate rows because the columns you're matching on have multiple potential matches. In the example below we're going to try an Inner Join. You'll notice the columns in Orange were duplicated.



Join duplication "error"

This isn't an error per se but it is something to watch out for as it can cause you to duplicate data you don't intend to duplicate.



Join duplication "error"

## UNION

Query:

```
SELECT * FROM dataset_1 UNION SELECT * FROM table_to_union
```

Result:

In order to see the effect of the result, let's run this query:

```
SELECT DISTINCT destination FROM ( SELECT * FROM dataset_1 UNION SELECT * FROM table_to_union);
```

ABC destination	
1	Home
2	No Urgent Place
3	UNION
4	Work

## JOIN (LEFT, INNER, OUTER, RIGHT)

Query:

```
SELECT a.destination, a.time, b.part_of_day FROM dataset_1 a INNER JOIN table
_to_join b ON a.time = b.time;
```

Result:

	ABC destination	ABC time	ABC part_of_day
1	No Urgent Place	2PM	Afternoon
2	No Urgent Place	10AM	Morning
3	No Urgent Place	10AM	Morning
4	No Urgent Place	2PM	Afternoon
5	No Urgent Place	2PM	Afternoon
6	No Urgent Place	6PM	Evening
7	No Urgent Place	2PM	Afternoon
8	No Urgent Place	10AM	Morning
9	No Urgent Place	10AM	Morning
10	No Urgent Place	10AM	Morning
11	No Urgent Place	2PM	Afternoon
12	No Urgent Place	2PM	Afternoon

## Advanced Querying

### Subquerying

Query:

```
SELECT destination, passanger FROM (SELECT * FROM dataset_1 WHERE passanger = 'Alone');
```

Result:

	ABC destination	ABC passanger
1	No Urgent Place	Alone
2	Home	Alone
3	Home	Alone
4	Home	Alone
5	Work	Alone
6	Work	Alone
7	Work	Alone
8	Work	Alone
9	Work	Alone
10	Work	Alone

## Advanced Filtering

LIKE

Query:

```
SELECT * FROM dataset_1 WHERE weather LIKE 'Sun%';
```

Result:

	RBC destination	RBC passanger	RBC weather	123 temperature	RBC time	RBC coupon
1	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)
2	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House
3	No Urgent Place	Friend(s)	Sunny	80	10AM	Carry out & Take away
4	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House
5	No Urgent Place	Friend(s)	Sunny	80	2PM	Coffee House
6	No Urgent Place	Friend(s)	Sunny	80	6PM	Restaurant(<20)
7	No Urgent Place	Friend(s)	Sunny	55	2PM	Carry out & Take away
8	No Urgent Place	Kid(s)	Sunny	80	10AM	Restaurant(<20)
9	No Urgent Place	Kid(s)	Sunny	80	10AM	Carry out & Take away
10	No Urgent Place	Kid(s)	Sunny	80	10AM	Bar
11	No Urgent Place	Kid(s)	Sunny	80	2PM	Restaurant(<20)
12	No Urgent Place	Kid(s)	Sunny	55	2PM	Restaurant(<20)
13	No Urgent Place	Kid(s)	Sunny	55	6PM	Coffee House
14	Home	Alone	Sunny	55	6PM	Bar
15	Home	Alone	Sunny	55	6PM	Restaurant(20-50)
16	Home	Alone	Sunny	80	6PM	Coffee House
17	Work	Alone	Sunny	55	7AM	Coffee House
18	Work	Alone	Sunny	55	7AM	Bar
19	Work	Alone	Sunny	80	7AM	Restaurant(20-50)
20	Work	Alone	Sunny	80	7AM	Carry out & Take away
21	Work	Alone	Sunny	55	7AM	Restaurant(<20)
22	Work	Alone	Sunny	55	7AM	Coffee House
23	No Urgent Place	Alone	Sunny	55	2PM	Restaurant(<20)
24	No Urgent Place	Friend(s)	Sunny	80	10AM	Coffee House
25	No Urgent Place	Friend(s)	Sunny	80	10AM	Bar

## Wildcard Operators

### Wildcard Characters in SQL Server

Symbol	Description	Example
%	Represents zero or more characters	bl% finds bl, black, blue, and blob
_	Represents a single character	h_t finds hot, hat, and hit
[]	Represents any single character within the brackets	h[oa]t finds hot and hat, but not hit
^	Represents any character not in the brackets	h[^oa]t finds hit, but not hot and hat
-	Represents a range of characters	c[a-b]t finds cat and cbt

[https://www.w3schools.com/sql/sql\\_wildcards.asp](https://www.w3schools.com/sql/sql_wildcards.asp)

## BETWEEN

Query:

```
SELECT DISTINCT temperature FROM dataset_1 WHERE temperature BETWEEN 29 AND 75;
```

Result:

	123 temperature
1	55
2	30

IN

Query:

```
SELECT occupation FROM dataset_1 WHERE occupation IN ('Sales & Related', 'Management');
```

Result:

	RBC occupation
1	Sales & Related
2	Sales & Related
3	Sales & Related
4	Sales & Related
5	Sales & Related
6	Sales & Related
7	Sales & Related
8	Sales & Related
9	Sales & Related
10	Sales & Related
11	Sales & Related
12	Sales & Related
13	Sales & Related
14	Sales & Related
15	Sales & Related
16	Sales & Related
17	Sales & Related
18	Sales & Related
19	Sales & Related
20	Sales & Related
21	Sales & Related
22	Sales & Related
23	Management
24	Management
25	Management