SQL for Data Analysts

What is SQL

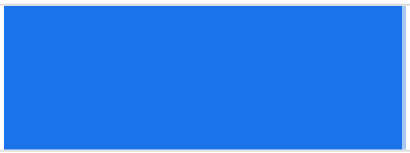
SQL (pronounced S.Q.L. or Sequel, no one really cares) stands for Structured Querying Language and is a language that is used to query, form, and manipulate databases.

Setting up the Environment

Files to Download

SQL Course Public Downloads Folder - Google Drive

https://drive.google.com/drive/folders/1t1l7mcnhxN0rvYH-WTSSSuwocyTu...



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, Sybase, MS Access, Teradata,

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

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 '@' cs.duke.edu, Duke University Data Set

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 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 - Created 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
- VARCHAR
- TEXT

Sarim Akbar Sep 5
Generally same length of number of characters of every variable in the column

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

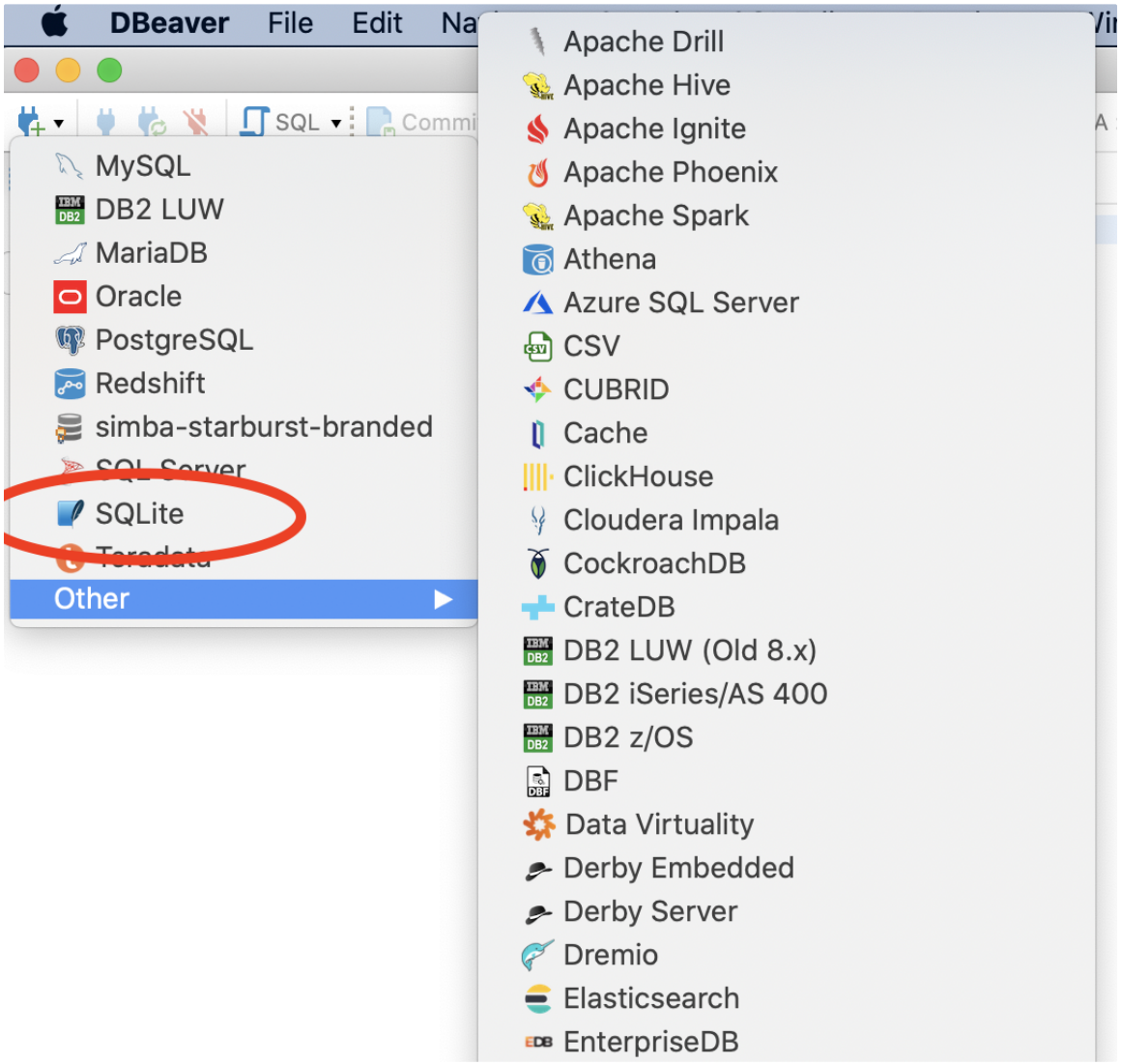
- TINYBLOB
- BLOB

S **Sarthak Agrawal** Jun 15
Variable character (upto 255 characters)

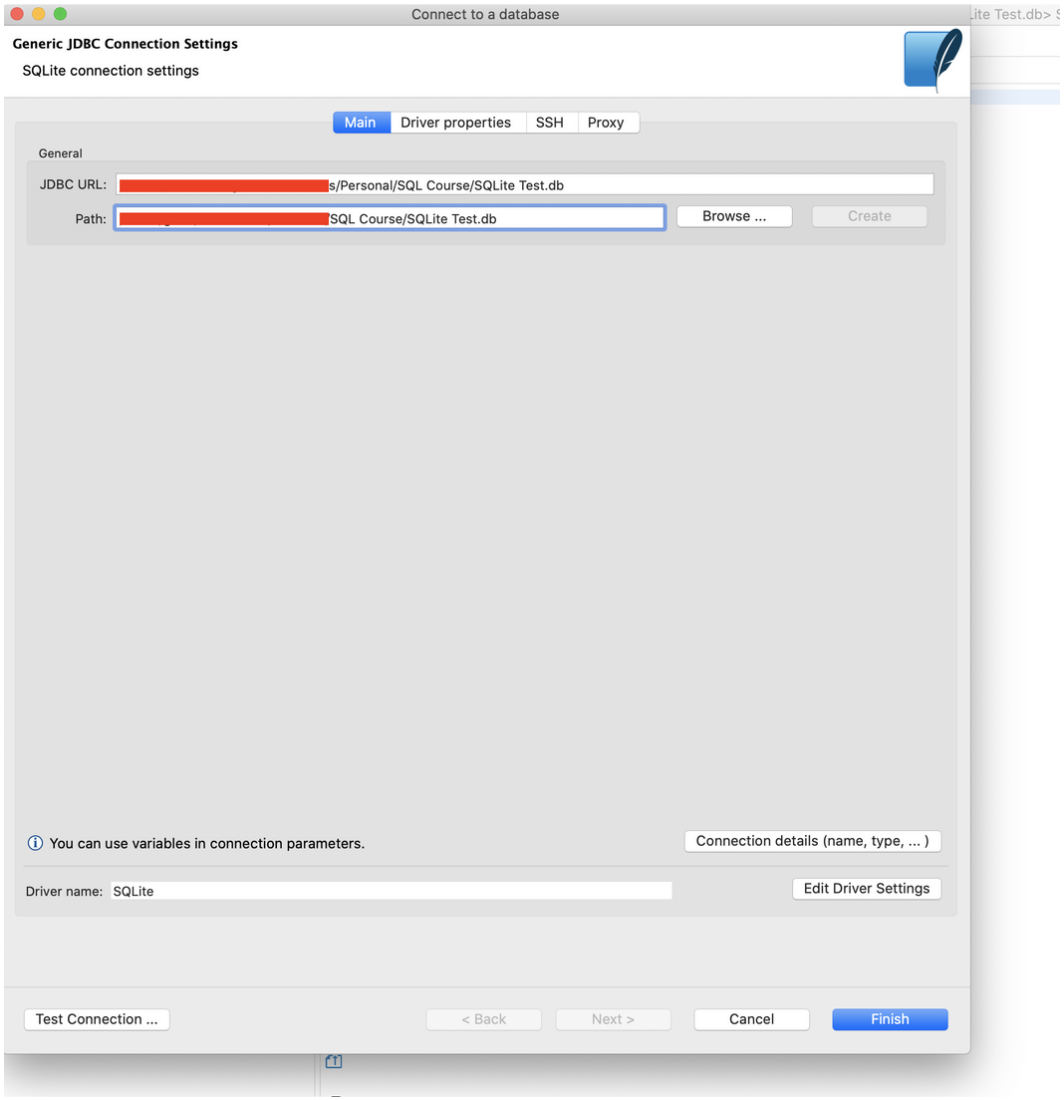
M **Manish Padole** Sep 29
It is an alias name for **varying** in SQL

Basic Querying

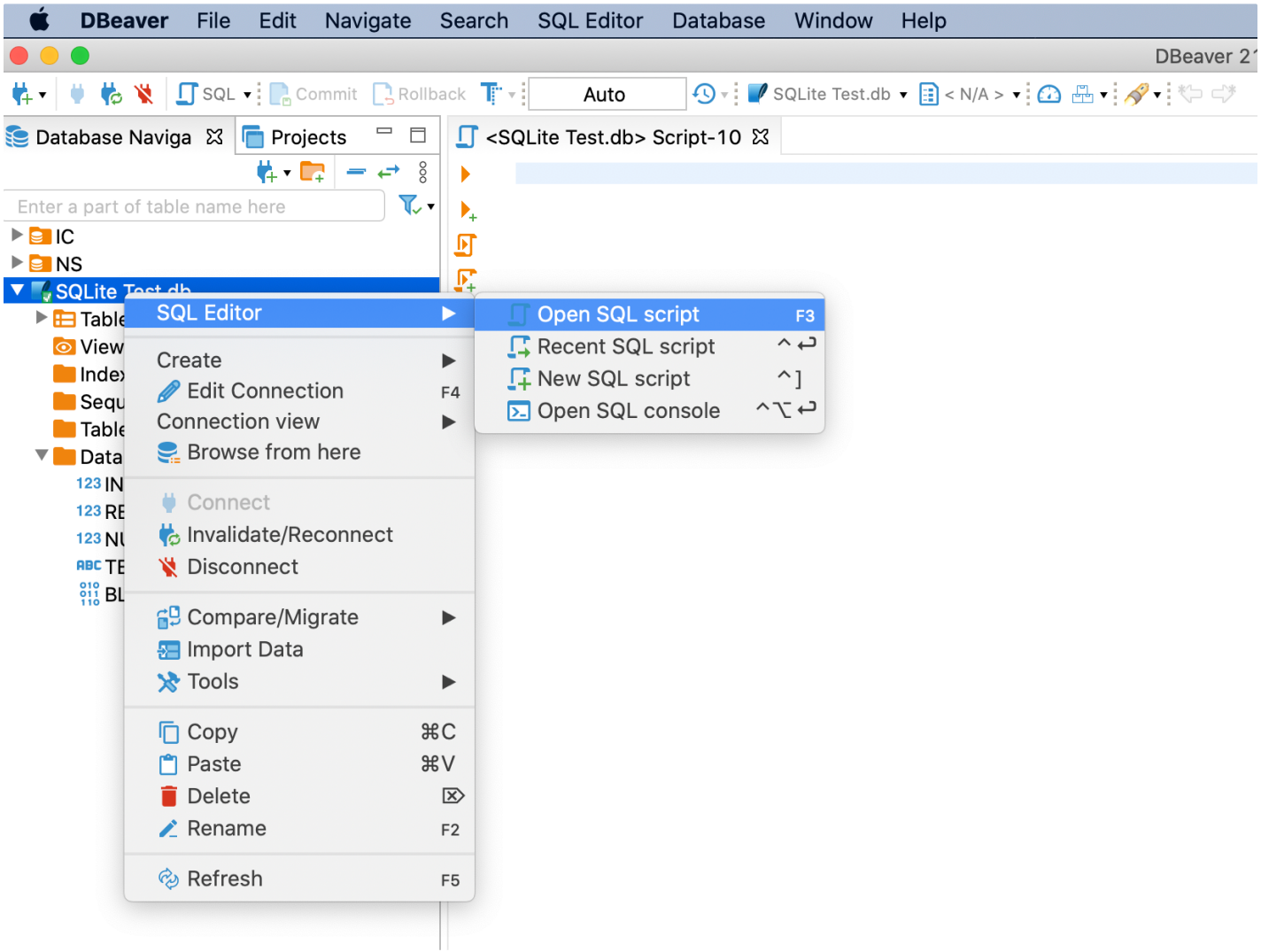
After opening DBeaver and downloading the database file from the link mentioned at the top of the page, create a new connection and select the option for SQLite.



Open up the database file you downloaded



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



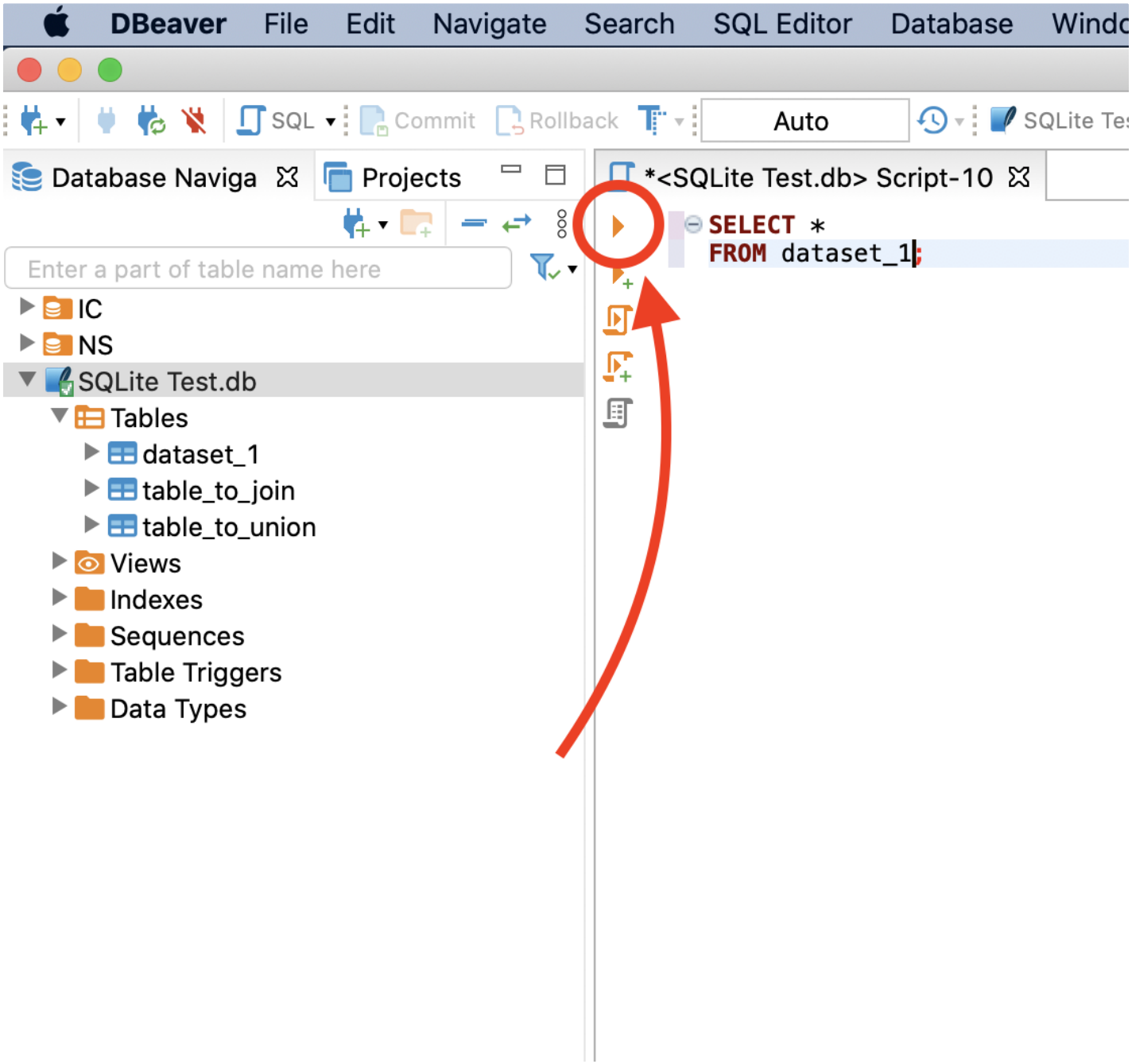
Obtaining Data

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


DBeaver File Edit Navigate Search SQL Editor Database Windo
SQL
Commit
Rollback
Auto
SQLite Tes
Database Naviga
Projects
*<SQLite Test.db> Script-10
SELECT *
FROM dataset_1;
Enter a part of table name here
IC
NS
SQLite Test.db
Tables
dataset_1
table_to_join
table_to_union
Views
Indexes
Sequences
Table Triggers
Data Types

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:

	destination	passanger	weather	temperature	time	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;
```

I've noticed that you cant use order by after using limit. you might put that in... more

Result:

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

WHERE

Query:

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

Result:

	destination	passanger	weather	temperature	time	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:

1

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

Result:

	destination	passanger	weather	temperature	time	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:

	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:

	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:

	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:

	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:

	weather	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 GROU
P BY weather;
```

Result:

	weather	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:

	weather	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:

	weather	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:

	weather	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:

	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.

Style	Color	Model
Big	Blue	30X
Small	Red	50X

Style	Color	Model	Car
Medium	Chartreuse	123H	Atoyot
Puny	Gamboge	313J	Drof

Union

Style	Color	Model	Car
Big	Blue	30X	*Null*
Small	Red	50X	*Null*
Medium	Chartreuse	123H	Atoyot
Puny	Gamboge	313J	Drof

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

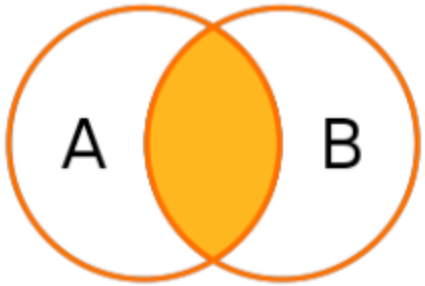
Left Table

ID	Car
1	Atoyot
2	Drof

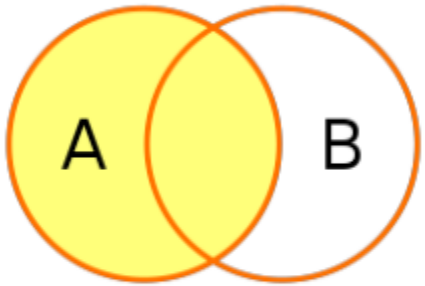
Right Table

There are four basic joins that you can use.

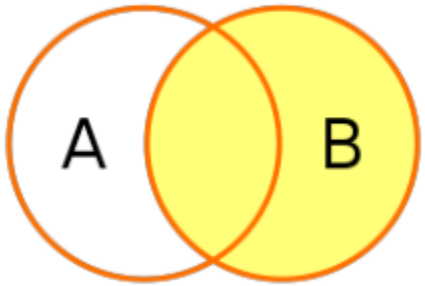
Inner Join



Left Join



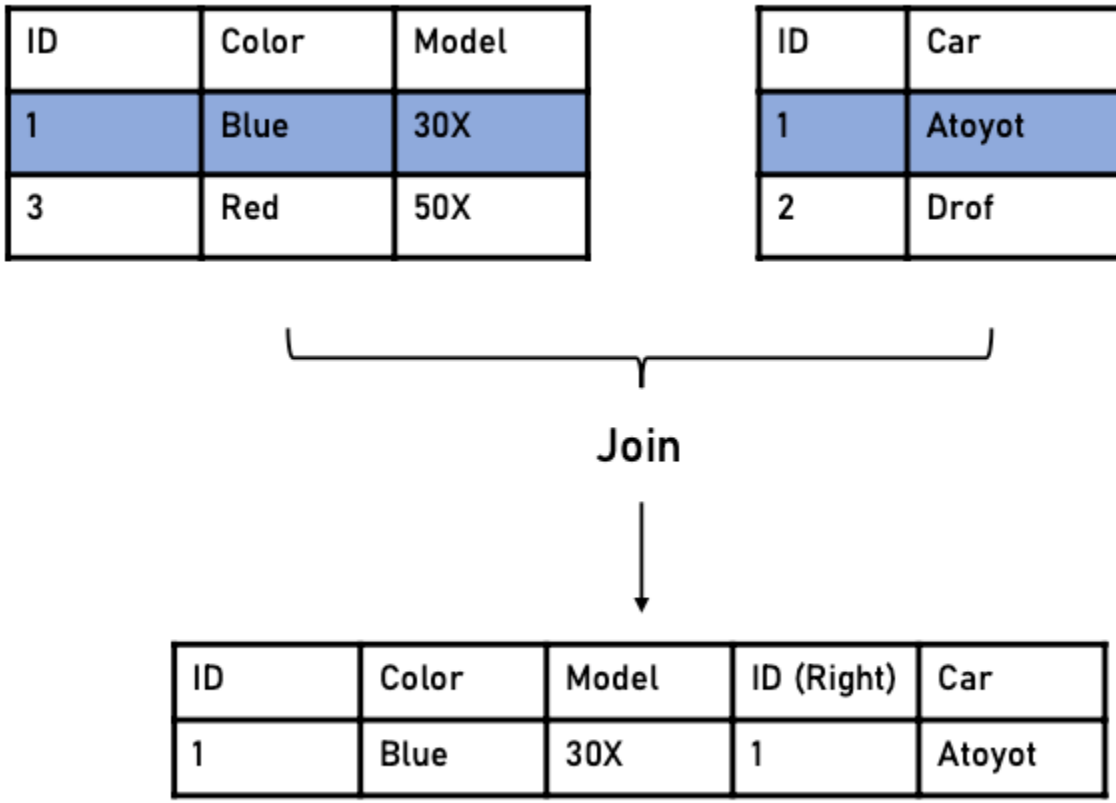
Right Join



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.

ID	Color	Model
1	Blue	30X
3	Red	50X

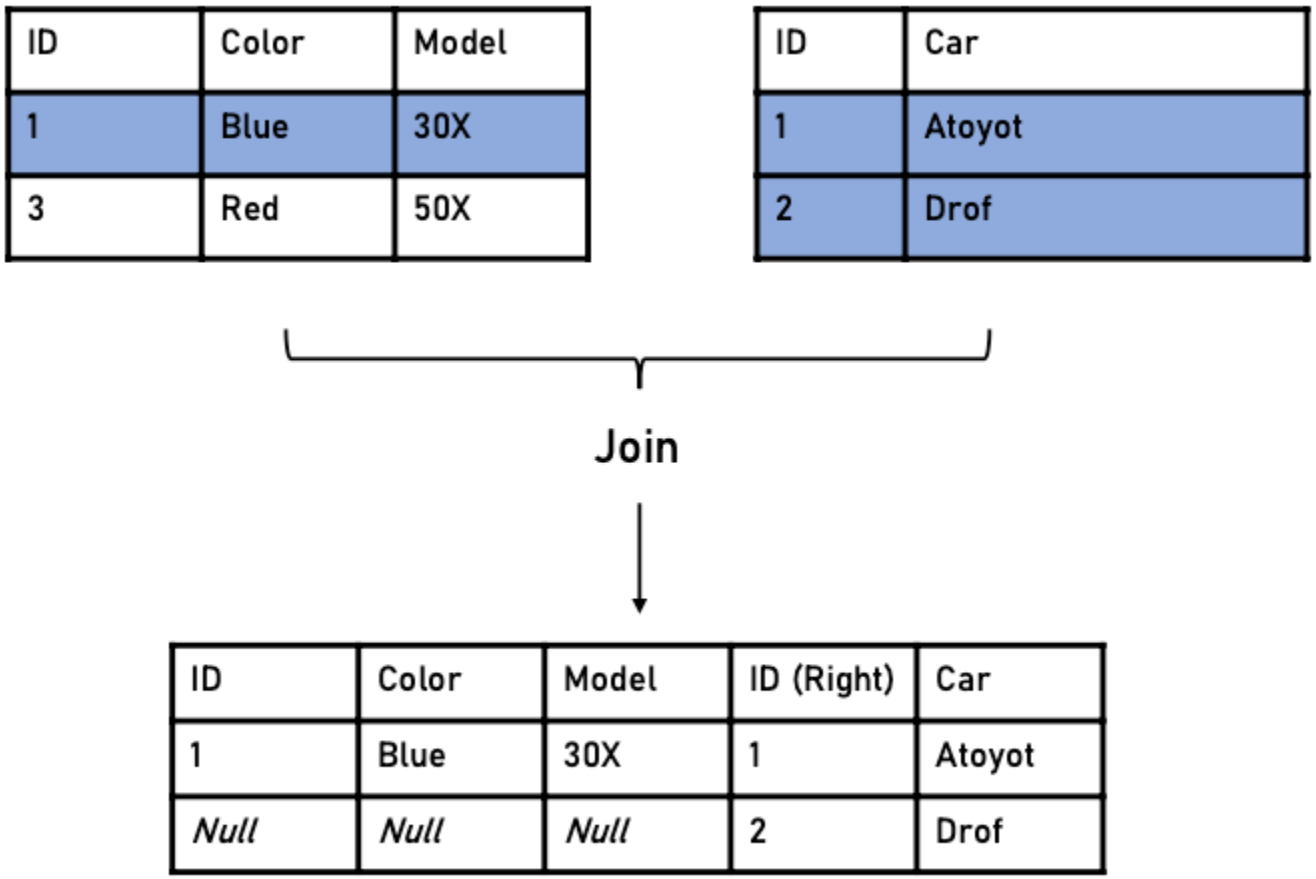
ID	Car
1	Atoyot
2	Drof

Join

ID	Color	Model	ID (Right)	Car
1	Blue	30X	1	Atoyot
3	Red	50X	*Null*	*Null*

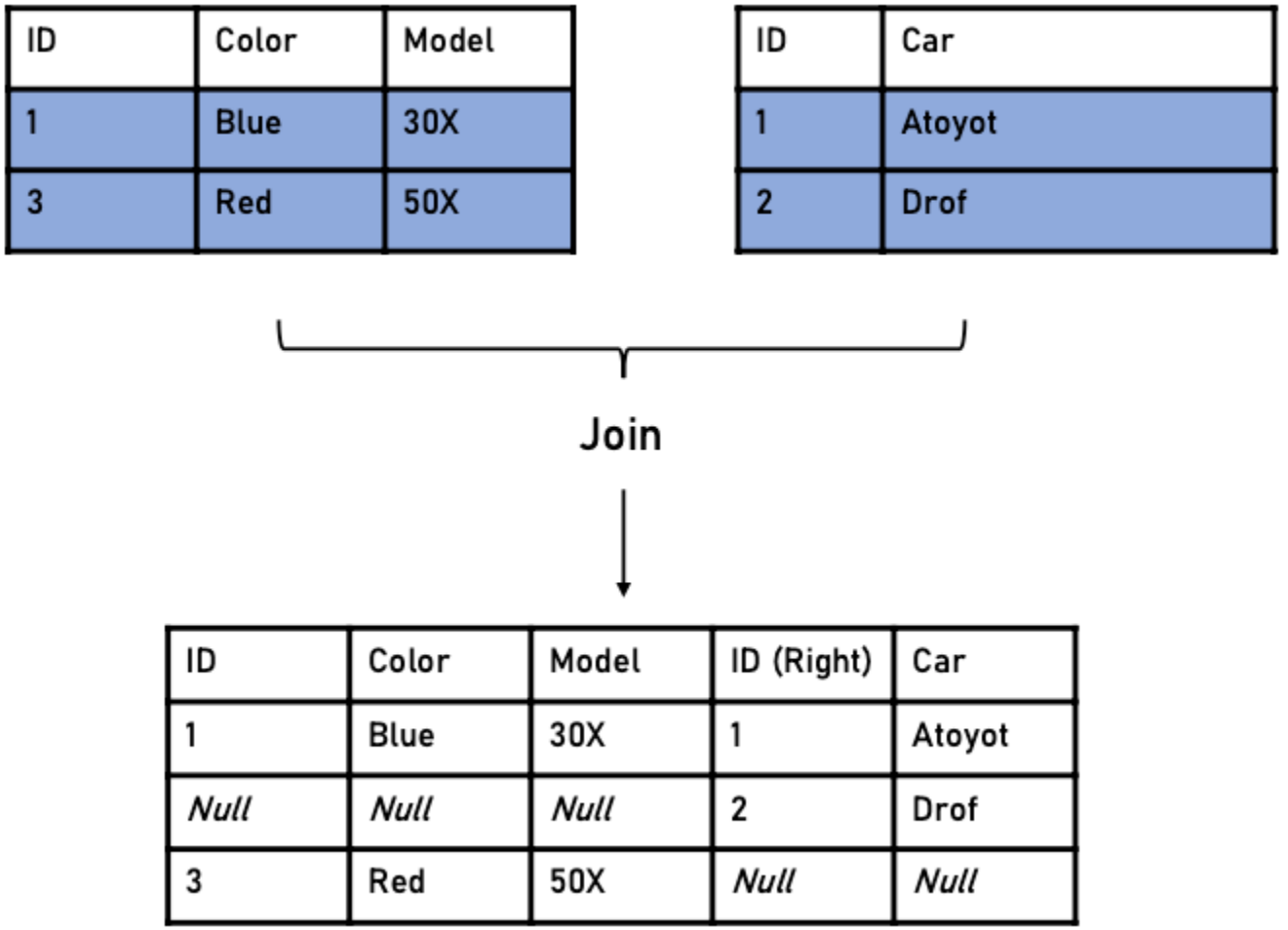
Left Join

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



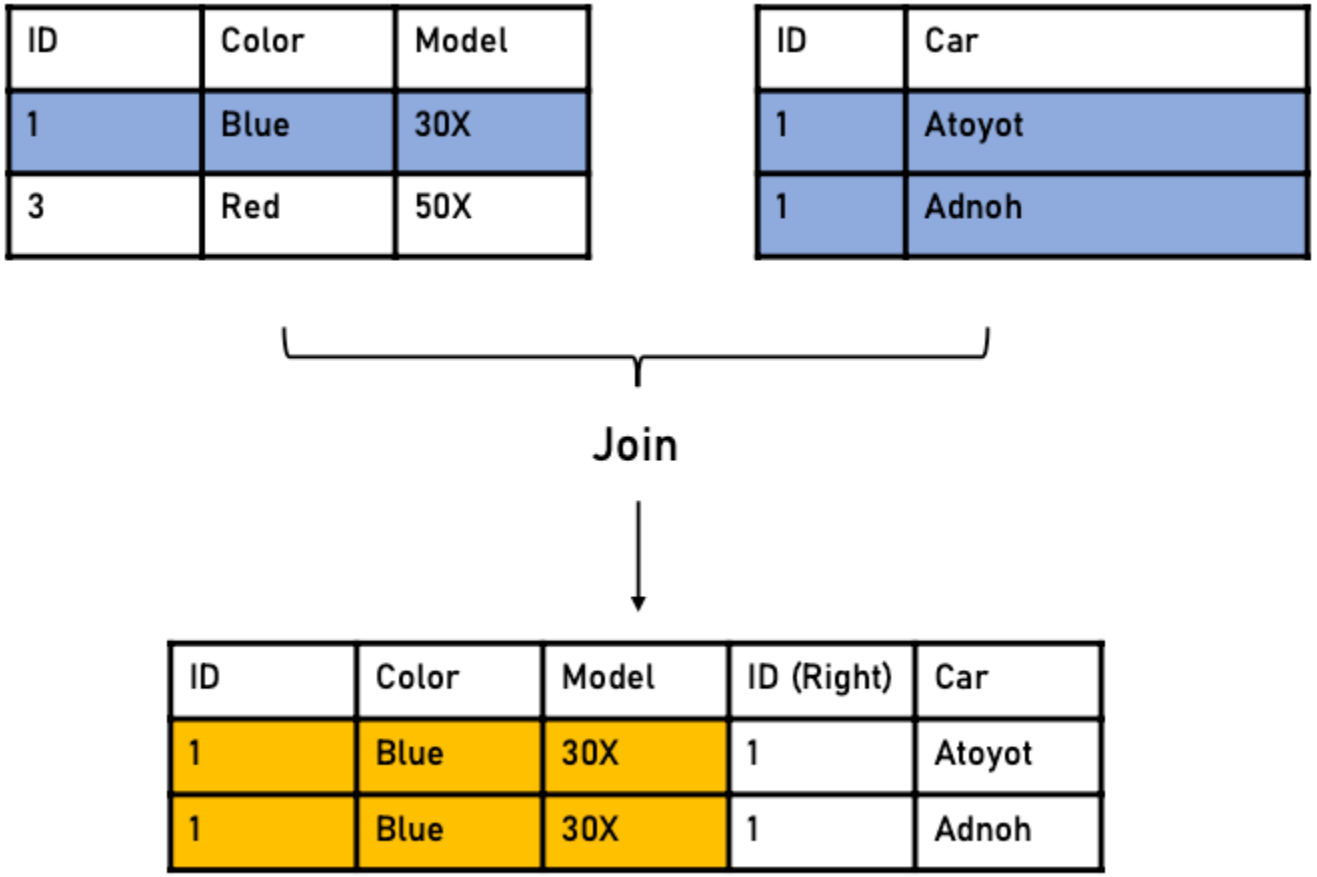
Right Join

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.

ID	Color	Model
1	Blue	30X
3	Red	50X

ID	Car
1	Atoyot
1	Adnoh

Join

ID	Color	Model	ID (Right)	Car
1	Blue	30X	1	Atoyot
1	Blue	30X	1	Adnoh

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:

	destination	time	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:

	destination	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:

	destination	passanger	weather	temperature	time	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

BETWEEN

Query:

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

Result:

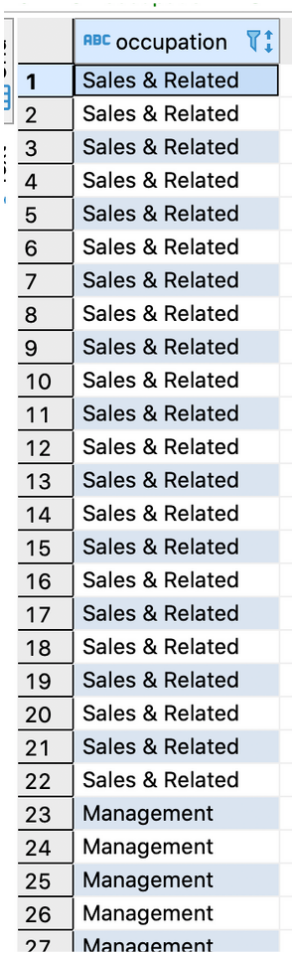
	123 temperature
1	55
2	30

IN

Query:

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

Result:



	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
26	Management
27	Management

Advanced Aggregation

Window Functions

OVER and PARTITION BY

Query:

```
SELECT destination, weather, AVG(temperature) OVER (PARTITION BY weather) AS 'avg_temp_by_weather' FROM dataset_1;
```

Result:

	destination	weather	avg_temp_by_weather
1429	Home	Snowy	30
1430	Work	Snowy	30
1431	Work	Snowy	30
1432	No Urgent Place	Snowy	30
1433	No Urgent Place	Snowy	30
1434	Home	Snowy	30
1435	Home	Snowy	30
1436	Work	Snowy	30
1437	Work	Snowy	30
1438	No Urgent Place	Snowy	30
1439	No Urgent Place	Snowy	30
1440	No Urgent Place	Snowy	30
1441	Home	Snowy	30
1442	Home	Snowy	30
1443	Work	Snowy	30
1444	Work	Snowy	30
1445	No Urgent Place	Snowy	30

You can also order by a certain column too if you want:

```
SELECT destination, weather, AVG(temperature) OVER (PARTITION BY weather ORDER BY destination) AS 'avg_temp_by_weather' FROM dataset_1;
```

ROW_NUMBER()

Query:

```
SELECT destination, weather, ROW_NUMBER() OVER (PARTITION BY weather ORDER BY destinatio
n) AS 'avg_temp_by_weather' FROM dataset_1;
```

Result:

	destination	weather	avg_temp_by_weather
1	Home	Rainy	1
2	Home	Rainy	2
3	Home	Rainy	3
4	Home	Rainy	4
5	Home	Rainy	5
6	Home	Rainy	6
7	Home	Rainy	7
8	Home	Rainy	8
9	Home	Rainy	9
10	Home	Rainy	10
11	Home	Rainy	11
12	Home	Rainy	12
13	Home	Rainy	13
14	Home	Rainy	14
15	Home	Rainy	15
16	Home	Rainy	16
17	Home	Rainy	17
18	Home	Rainy	18
19	Home	Rainy	19
20	Home	Rainy	20
21	Home	Rainy	21
22	Home	Rainy	22
23	Home	Rainy	23
24	Home	Rainy	24
25	Home	Rainy	25
26	Home	Rainy	26
27	Home	Rainy	27
28	Home	Rainy	28
29	Home	Rainy	29
30	Home	Rainy	30
31	Home	Rainy	31
32	Home	Rainy	32
33	Home	Rainy	33

RANK()

Query:

```
SELECT destination, weather, RANK() OVER (PARTITION BY weather ORDER BY destination) AS
'avg_temp_by_weather' FROM dataset_1;
```

Result:

	destination	weather	avg_temp_by_weather
250	Home	Rainy	1
251	Home	Rainy	1
252	Home	Rainy	1
253	Home	Rainy	1
254	Home	Rainy	1
255	No Urgent Place	Rainy	255
256	No Urgent Place	Rainy	255
257	No Urgent Place	Rainy	255
258	No Urgent Place	Rainy	255
259	No Urgent Place	Rainy	255
260	No Urgent Place	Rainy	255
261	No Urgent Place	Rainy	255
262	No Urgent Place	Rainy	255
263	No Urgent Place	Rainy	255

DENSE_RANK()

Query:

```
SELECT destination, weather, DENSE_RANK() OVER (PARTITION BY weather ORDER BY destinatio
n) AS 'avg_temp_by_weather' FROM dataset_1;
```

Result:

	destination	weather	avg_temp_by_weather
253	Home	Rainy	1
254	Home	Rainy	1
255	No Urgent Place	Rainy	2
256	No Urgent Place	Rainy	2
257	No Urgent Place	Rainy	2
258	No Urgent Place	Rainy	2
259	No Urgent Place	Rainy	2
260	No Urgent Place	Rainy	2
261	No Urgent Place	Rainy	2
262	No Urgent Place	Rainy	2
263	No Urgent Place	Rainy	2
264	No Urgent Place	Rainy	2

NTILE

Query:

```
SELECT time, NTILE (50) OVER (ORDER BY time) FROM dataset_1;
```

Result:

	time	NTILE (50) OVER (ORDER BY time)
251	10AM	1
252	10AM	1
253	10AM	1
254	10AM	1
255	10AM	2
256	10AM	2
257	10AM	2
258	10AM	2
259	10AM	2
260	10AM	2
261	10AM	2
262	10AM	2
263	10AM	2
264	10AM	2
265	10AM	2
266	10AM	2

LAG

Query:

```
SELECT destination, time, LAG(row_count , 1, '99999') OVER (ORDER BY row_count) AS 'Lagg
edCount' FROM dataset_1;
```

Result:

	destination	time	LaggedCount
1	No Urgent Place	2PM	99999
2	No Urgent Place	10AM	1
3	No Urgent Place	10AM	2
4	No Urgent Place	2PM	3
5	No Urgent Place	2PM	4
6	No Urgent Place	6PM	5
7	No Urgent Place	2PM	6
8	No Urgent Place	10AM	7
9	No Urgent Place	10AM	8
10	No Urgent Place	10AM	9
11	No Urgent Place	2PM	10

LEAD

Query:

```
SELECT destination, time, LEAD(row_count , 1, '99999') OVER (ORDER BY row_count) AS 'Lag
gedCount' FROM dataset_1;
```

Result:

	destination	time	LaggedCount
1	No Urgent Place	2PM	2
2	No Urgent Place	10AM	3
3	No Urgent Place	10AM	4
4	No Urgent Place	2PM	5
5	No Urgent Place	2PM	6
6	No Urgent Place	6PM	7
7	No Urgent Place	2PM	8
8	No Urgent Place	10AM	9
9	No Urgent Place	10AM	10
10	No Urgent Place	10AM	11
11	No Urgent Place	2PM	12
12	No Urgent Place	2PM	13
13	No Urgent Place	6PM	14
14	Home	6PM	15
15	Home	6PM	16
16	Home	6PM	17
17	Work	7AM	18
18	Work	7AM	19

Manipulating Tables

Basics

CREATE

DROP

ALTER

Views

Tips

Full Query Example

Further Research

Stored Procedures → Stored Procs

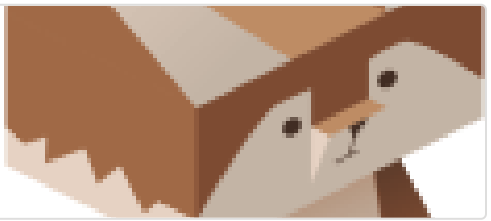
Query Optimization

> Performance Tuning SQL Queries | Advanced SQL - Mode Analytics
>
> Starting here? This lesson is part of a full-length tutorial in using SQL for Data Analysis. Check out the beginning. In this lesson we'll cover: The lesson on subqueries introduced the idea that you can sometimes create the same desired result set with a
>
> https://mode.com/sql-tutorial/sql-performance-tuning/

References

Basic SQL Reference

> SQL Tutorial
>
> SQL is a standard language for storing, manipulating and retrieving data in databases. Our SQL tutorial will teach you how to use SQL in: MySQL, SQL Server,
>
> https://www.w3schools.com/sql/

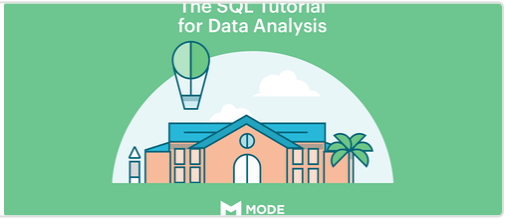


Window Functions

SQL Window Functions | Advanced SQL - Mode Analytics

Starting here? This lesson is part of a full-length tutorial in using SQL for Data Analysis. Check out the beginning. In this lesson we'll cover: This lesson uses data

https://mode.com/sql-tutorial/sql-window-functions/



Database Schema Information

What are database schemas? 5 minute guide with examples

A database schema is an abstract design that represents the storage of your data in a database. Learn the basics of database schemas with common examples:

https://www.educative.io/blog/what-are-database-schemas-examples