SQL Workshop

SQL 101

Iyed Ghedamsi

2021-12-09

Software tools

SQL and RDBMS

Srtuctured **Q**uery **L**anguage

The language used to deal with data and databases.

It is standardised in **ISO SQL:2006**

SQL and RDBMS

Srtuctured **Q**uery **L**anguage

The language used to deal with data and databases.

It is standardised in **ISO SQL:2006**

Relational **D**atabase **M**anagement **S**ystem

The software implementing the language. It is responsible for executing the queries.

There are multiple implementations of the **SQL** standard.

Relational Database Management Systems

Relational Database Management Systems

Syntax differences

- Case sensitivity
- Quotation marks
- Date functions
- Window functions

Relational Database Management Systems

Syntax differences

- Case sensitivity
- Quotation marks
- Date functions
- Window functions

Implementation differences

Has to do with **How** queries are run. Usually of importance to System Administrators.

Which one to use ?

SQLite

- Embedded devices
- Low traffic websites
- Data analysis
- Data transfer format
- Education and training

Which one to use ?

SQLite

- Embedded devices
- Low traffic websites
- Data analysis
- Data transfer format
- Education and training

Client/Server RDMBS (PostgreSQL)

- Client / Server applications (over a network)
- High-volume Websites
- Very large datasets
- High concurrency

SQL Clients

Use shell clients

```
~
❯ psql
psql (13.5 (Ubuntu 13.5-0ubuntu0.21.04.1))
Type "help" for help.

iyed=# \c
You are now connected to database "iyed" as user "iyed".
iyed=#
```

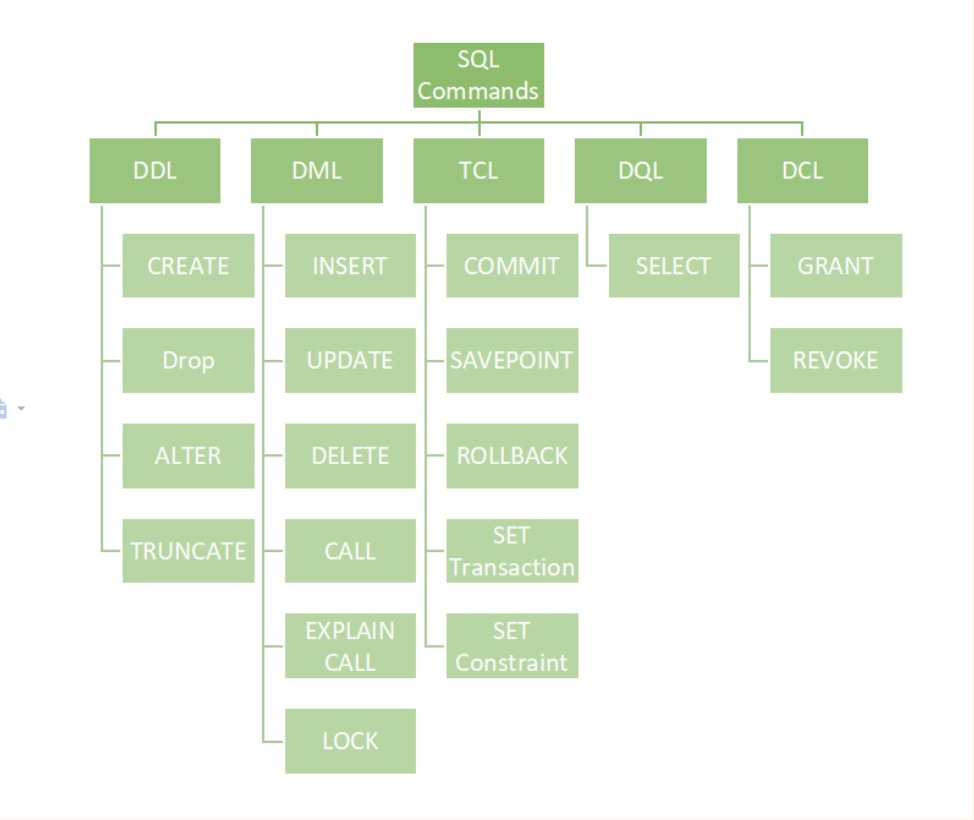
PostgreSQL Shell

```
~
❯ sqlite3
SQLite version 3.34.1 2021-01-20 14:10:07
Enter ".help" for usage hints.
Connected to a transient in-memory database.
Use ".open FILENAME" to reopen on a persistent database.
sqlite>
```

GUI Clients

DBeaver

SQL Sublanguages



Source: https://www.geeksforgeeks.org/sql-ddl-dql-dml-dcl-tcl-commands/

Intermission: Make sure software is installed.

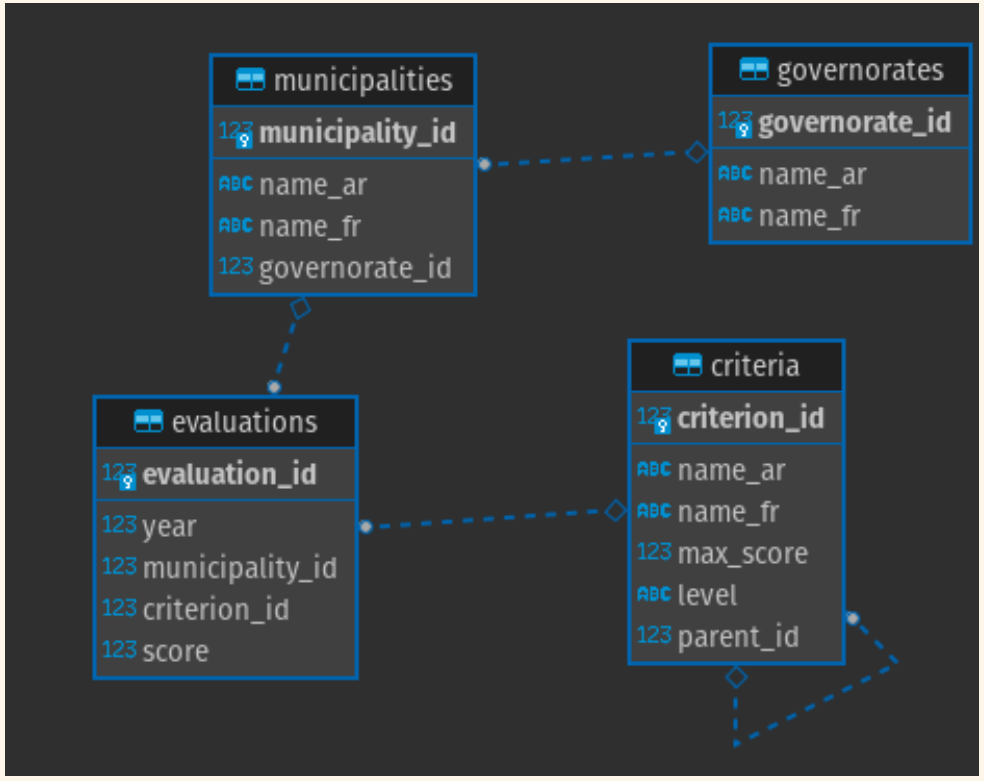
```
SELECT "Hello World!";
```

SEE YOU SPACE COWBOY

Querying SQL data

- Use a **SELECT** Statement to Retrieve Data
- Use the **WHERE** Clause to Define Search Conditions
- Use the **GROUP BY** Clause to Group Query Results
- Use the **HAVING** Clause to Specify Group Search Conditions
- Use the **ORDER BY** Clause to Sort Query Results

Meet our database



Using a SELECT statement.

Syntax:

```
SELECT [ DISTINCT | ALL ] { * | <select list> }
FROM <table reference> [ { , <table reference> } . . . ]
```

Using a SELECT statement.

Syntax:

```
SELECT [ DISTINCT | ALL ] { * | <select list> }
FROM <table reference> [ { , <table reference> } . . . ]
```

```
SELECT *
  FROM municipalities;
```

municipality_id	name_ar	name_fr	governorate_id
1111	تونس العاصمة	Tunis Ville	11
1112	باردو	Le Bardo	11
1113	الكرم	EL kram	11
1114	حلق الوادي	La Goulette	11
1115	قرطاج	Carthage	11
1116	سيدي بوسعيد	Sidi Bou Said	11

Using the WHERE clause for search conditions.

Syntax | SQL | Output

```
SELECT [ DISTINCT | ALL ] { * | <select list> }
FROM <table reference> [ { , <table reference> } . . . ]
[ WHERE <search condition> ]
```

The **WHERE** clause represents a filter for the result returned by the **FROM** clause.

Using the WHERE clause for search conditions.

Syntax | SQL | Output

```sql
SELECT *
FROM municipalities
WHERE municipalities.governorate_id = 11;
```

Using the WHERE clause for search conditions.

Syntax SQL Output

municipality_id	name_ar	name_fr	governorate_id
1111	تونس العاصمة	Tunis Ville	11
1112	باردو	Le Bardo	11
1113	الكرم	EL kram	11
1114	حلق الوادي	La Goulette	11
1115	قرطاج	Carthage	11
1116	سيدي بوسعيد	Sidi Bou Said	11
1117	المرسى	La Marsa	11
1118	سيدي حسين	Sidi Hassine	11

Using multiple predicates with the WHERE clause

OR

	True	False	NULL
True	True	True	True
False	True	False	NULL
NULL	True	NULL	NULL

AND

	True	False	NULL
True	True	False	NULL
False	False	False	False
NULL	NULL	False	NULL

Using multiple predicates with the WHERE clause (*Cont.*)

SQL | Output | SQL multiple predicates | Output multiple predicates

```sql
SELECT *
FROM municipalities
WHERE
  municipalities.governorate_id >= 11 AND
  municipalities.governorate_id <= 13;
```

Using multiple predicates with the WHERE clause (*Cont.*)

SQL | Output | SQL multiple predicates | Output multiple predicates

municipality_id	name_ar	name_fr	governorate_id
1313	حمام الأنف	Hammam Lif	13
1112	باردو	Le Bardo	11
1212	سكرة	Soukra	12
1316	الزهراء	Ez-Zahra	13
1314	حمام الشط	Hammam Chott	13

Using multiple predicates with the WHERE clause (*Cont.*)

SQL | Output | **SQL multiple predicates** | Output multiple predicates

```sql
SELECT *
FROM municipalities AS m
WHERE
  m.name_fr LIKE "L%";
```

Using multiple predicates with the WHERE clause (*Cont.*)

SQL | Output | SQL multiple predicates | **Output multiple predicates**

municipality_id	name_ar	name_fr	governorate_id
1112	باردو	Le Bardo	11
1114	حلق الوادي	La Goulette	11
1117	المرسى	La Marsa	11
2311	الكاف	Le Kef	23
3239	لمطة	Lamta	32
6121	لالة	Lala	61

Other comparison operators

- **NOT**: used to negate the return value of a predicate.
- **<> OR !=** (not equal): used to accept any value that is **different** from the specified value for this operator.

SQL | Output

```
SELECT DISTINCT municipalities.governorate_id
FROM municipalities
WHERE municipalities.governorate_id != 11;
```

Other comparison operators

- **NOT**: used to negate the return value of a predicate.
- **<> OR !=** (not equal): used to accept any value that is **different** from the specified value for this operator.

SQL | Output

governorate_id
12
15
31
61
17
33

Practice problems

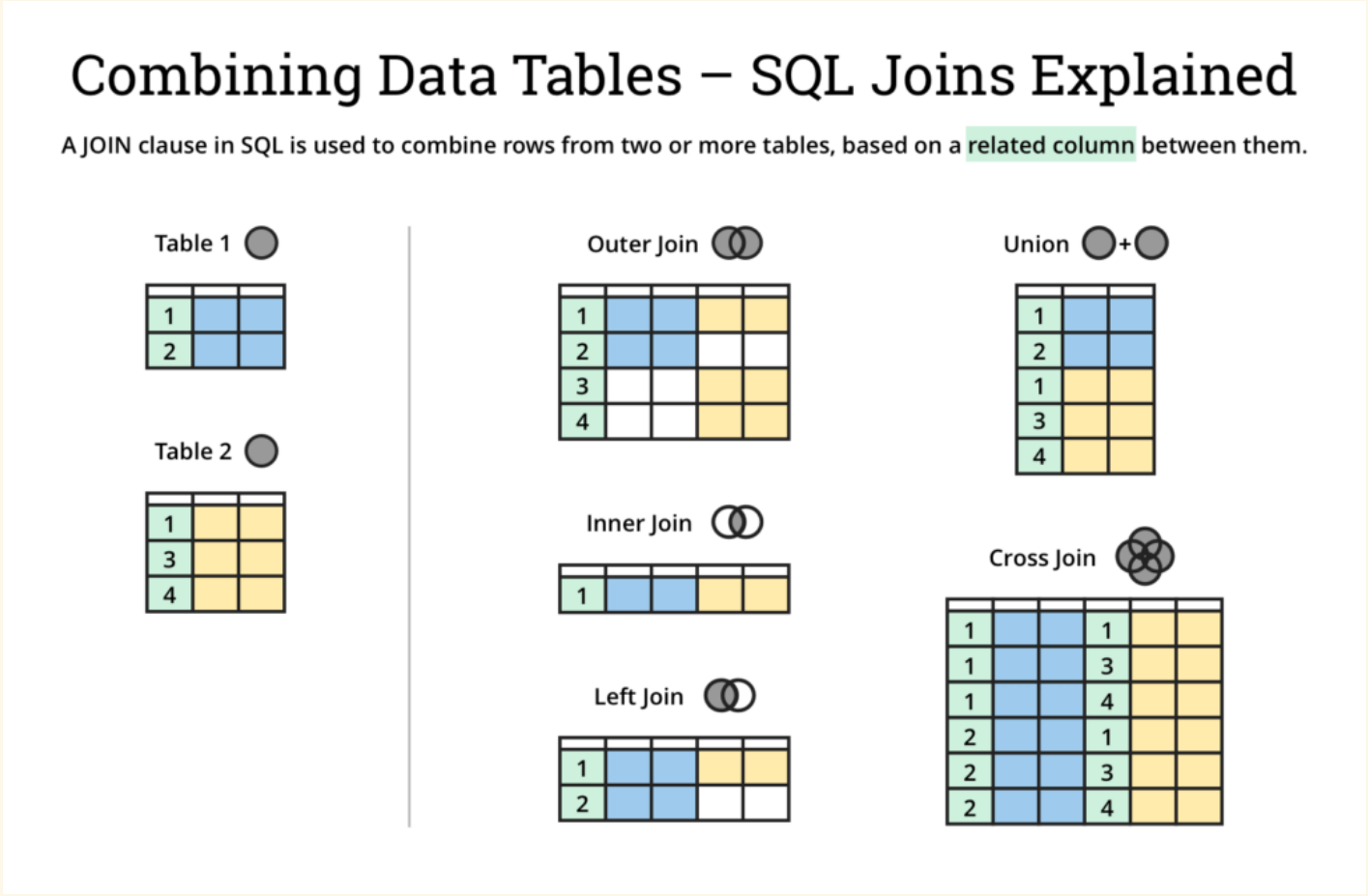
1. Show a list of municipality names in French and Arabic where the first letter of the French name is `F` .
2. Show a list of municipalities that are either in the governorate with `id = 11` or with `id = 41`. Their names do **NOT** start with the letter `T`.

Querying Multiple Tables

Key Skills & Concepts:

- Perform Basic Join Operations
- Join Tables with Shared Column Names
- Use the Condition Join
- Perform Union Operations

SQL Joins



Source: https://dataschool.com/how-to-teach-people-sql/sql-join-types-explained-visually/

Cross Joins

Also called cartesian product

SQL | Output (partial)

```
SELECT municipalities.municipality_id, governorates.governorate_id
FROM municipalities, governorates;
```

Cross Joins

Also called cartesian product

SQL | Output (partial)

municipality_id	governorate_id
1211	12
1211	21
1211	13
1211	17
1211	51
1211	61

Inner Joins

Inner joins are assumed by default. Using **JOIN** is equivalent to **INNER JOIN**

SQL | Output (partial)

```sql
SELECT
  m.municipality_id,
  m.name_fr,
  g.governorate_id,
  g.name_fr
FROM
  municipalities m
INNER JOIN
  governorates g
  ON
  m.governorate_id = g.governorate_id
WHERE
  g.governorate_id = 13;
```

Inner Joins

Inner joins are assumed by default. Using **JOIN** is equivalent to **INNER JOIN**

SQL | Output (partial)

municipality_id	name_fr	governorate_id	name_fr
1311	Ben Arous	13	Ben Arous
1312	El Mourouj	13	Ben Arous
1313	Hammam Lif	13	Ben Arous
1314	Hammam Chott	13	Ben Arous
1315	Bou Mhel El Bassatine	13	Ben Arous
1316	Ez-Zahra	13	Ben Arous

Left join

With a left join, we make sure we keep all records from the left most table in our query.

SQL | Output (partial)

```sql
SELECT m.name_fr, e.score
FROM   municipalities m
       LEFT JOIN evaluations e
              ON e.municipality_id = m.municipality_id
WHERE e.year = 2017;
```

Left join

With a left join, we make sure we keep all records from the left most table in our query.

SQL | Output (partial)

name_fr	score
Tunis Ville	22
Tunis Ville	21
Tunis Ville	38
Tunis Ville	4
Tunis Ville	8
Tunis Ville	10

Self join

Self joins are useful when a table contains a hierarchical or a graph structure.

SQL | Output (partial)

```
SELECT
    c.criterion_id
    , c.parent_id
    , pc.criterion_id AS pc_criterion_id
FROM
    criteria c
LEFT JOIN criteria pc ON
    c.parent_id = pc.criterion_id;
```

Self join

Self joins are useful when a table contains a hierarchical or a graph structure.

SQL | Output (partial)

criterion_id	parent_id	pc_criterion_id
10	NA	NA
11	NA	NA
12	NA	NA
101	10	10
102	10	10
103	10	10

Querying more than two tables

We wish to have a table showing a complete list of municipalities, their corresponding governorates, their evaluation criteria and scores for the year 2017.

Querying more than two tables

We wish to have a table showing a complete list of municipalities, their corresponding governorates, their evaluation criteria and scores for the year 2017.

```
SELECT
    m.name_fr municipality_name
    ,
    g.name_fr governorate_name
    ,
    c.name_fr criterion_name
    ,
    e.score
FROM
    municipalities m
LEFT JOIN evaluations e ON
    m.municipality_id = e.municipality_id
INNER JOIN governorates g ON
    m.governorate_id = g.governorate_id
INNER JOIN criteria c ON
    e.criterion_id = c.criterion_id
WHERE
    e."year" = 2017;
```

Practice problems:

1. Modify the last query to include the results only for criteria that do not have a parent.
2. Using a cross join, simulate the result of an inner join between the `municipalities` and `governorates` tables.
3. Which type of condition join should you use if you want to return only matched rows ?
4. How many tables are contained in a self-join?

Modifying SQL Data

page 195 **Key Skills & Concepts:**

- Insert SQL Data
- Update SQL Data
- Delete SQL Data

Inserting data

Syntax | SQL

```
INSERT INTO <table name>
[ ( <column name> [ { , <column name> } . . . ] ) ]
VALUES ( <value> [ { , <value> } . . . ] )
```

Inserting data

Syntax | SQL

```
INSERT INTO municipalities VALUES (0, 'TBS', 'TBS', 41);
INSERT INTO governorates (name_ar, name_fr) VALUES ("TBS GOV", "TBS GOV");

# This query is used to get the ID of the new governorate
SELECT * FROM governorates g WHERE g.name_ar = "TBS GOV";
```

Updating data

Syntax | SQL

```
UPDATE <table name>
SET <set clause expression> [ { , <set clause expression> } . . . ]
[ WHERE <search condition> ]
```

Updating data

Syntax | SQL

```sql
UPDATE
    municipalities
SET
    governorate_id = 64
WHERE
    governorate_id = 41
    AND name_ar = "TBS";
```

Deleting data

Syntax | SQL

```
DELETE FROM <table name>
[ WHERE <search condition> ]
```

Deleting data

Syntax | SQL

```sql
DELETE
FROM
    municipalities
WHERE
    municipality_id = 0;

DELETE
FROM
    governorates
WHERE
    name_fr = "TBS GOV";
```

Practice problems

1. Which SQL statement should you use to add data to a table?

    - (A) SELECT
    - (B) INSERT
    - (C) UPDATE
    - (D) DELETE

2. Which statement should you use to modify existing data in one or more rows in a table?

    - (A) SELECT
    - (B) INSERT
    - (C) UPDATE
    - (D) DELETE

3. Which statement or clause do you use in a DELETE statement to specify which rows are deleted from a table?

SQL Functions and Value Expressions

page 244

- Use Set Functions
- Use Value Functions
- Use Value Expressions
- Use Special Values

Set functions

I discuss five set functions: **COUNT**, **MAX**, **MIN**, **SUM**, and **AVG**

COUNT | COUNT (continued) | MAX/MIN | SUM | AVG

Count the total number of evaluations tracked in the database.

```
SELECT count(*) AS TOTAL_EVALUATIONS
FROM evaluations;
```

Set functions

I discuss five set functions: **COUNT**, **MAX**, **MIN**, **SUM**, and **AVG**

COUNT | COUNT (continued) | MAX/MIN | SUM | AVG

```
SELECT
    count(parent_id) AS count_of_parents
    , count(criterion_id) AS count_of_leaves
FROM
    criteria c;
```

Set functions

I discuss five set functions: **COUNT**, **MAX**, **MIN**, **SUM**, and **AVG**

COUNT | COUNT (continued) | MAX/MIN | SUM | AVG

```
SELECT
    e.criterion_id
    ,
    MAX(e.score) AS highest_score
    , MIN(e.score) AS lowest_score
FROM
    evaluations e
GROUP BY
    e.criterion_id ;
```

The **MAX** and **MIN** functions are not limited to numeric data. You can also use them to compare character strings.

Set functions

I discuss five set functions: **COUNT**, **MAX**, **MIN**, **SUM**, and **AVG**

COUNT | COUNT (continued) | MAX/MIN | SUM | AVG

```
SELECT
    e.municipality_id
    , SUM(e.score) AS TOTAL_SCORE
FROM
    evaluations e
WHERE
    e."year" = 2018
GROUP BY
    e.municipality_id
ORDER BY
    TOTAL_SCORE DESC;
```

Set functions

I discuss five set functions: **COUNT**, **MAX**, **MIN**, **SUM**, and **AVG**

COUNT | COUNT (continued) | MAX/MIN | SUM | AVG

```
SELECT
    e.municipality_id
    , AVG(e.score) AS TOTAL_SCORE
FROM
    evaluations e
WHERE
    e."year" = 2018
GROUP BY
    e.municipality_id
ORDER BY
    TOTAL_SCORE DESC;
```

Value functions

String value functions

SUBSTRING, **UPPER**, and **LOWER**

UPPER/LOWER | SUBSTRING

```
SELECT
    name_fr
    ,
    UPPER(name_fr)
    , LOWER(name_fr)
FROM
    governorates g;
```

String value functions

SUBSTRING, **UPPER**, and **LOWER**

UPPER/LOWER | SUBSTRING

```
SELECT
    name_fr AS municipality_name
    ,
    SUBSTRING(name_fr, 4, 100)
FROM
    municipalities m
WHERE
    m.name_fr LIKE "El%";
```

Datetime Value Functions

```
SELECT CURRENT_DATE;
SELECT CURRENT_TIME;
SELECT CURRENT_TIMESTAMP;
```

Using a CAST expression

To use the CAST value expression, you must specify the CAST keyword, and, in parentheses, provide the column name, the AS keyword, and the new data type.

```
SELECT
    CAST(
        SUBSTRING(criterion_id, 1, 2) AS INT
    ) AS INT_PARENT_ID
FROM
    evaluations e;
```

Practice problems

1. How many municipalities are there in Tunisia ?
2. How many municipalities have been evaluated in 2017 ?
3. What is the average score for a municipality in `Kairouan` in 2018 ?
4. Which set function should you use to add together the values in a column?
    - (A) MAX
    - (B) COUNT
    - (C) SUM
    - (D) AVG

Creating and Altering Tables

- Create SQL Tables
- Specify Column Data Types
- Specify Column Default Values
- Alter SQL Tables
- Delete SQL Tables

Creating tables

Syntax | SQL

```
CREATE TABLE <table name>
( <table element> [ { , <table element> } . . . ] )
```

Creating tables

Syntax | SQL

```
CREATE TABLE IF NOT EXISTS dsc_table
(
    trainee_id INTEGER
    ,
  trainee_first_name CHAR
    ,
  trainee_last_name CHAR
)
```

Drop SQL tables

Syntax | SQL

```
DROP TABLE <table name>;
```

Drop SQL tables

Syntax | SQL

```
DROP TABLE dsc_table;
```

Alter SQL tables

Syntax | SQL (Setup) | Adding a column

```
ALTER TABLE <table name>
ADD [COLUMN] <column definition>
| ALTER [COLUMN] <column name>
{ SET DEFAULT <default value> | DROP DEFAULT }
| DROP [COLUMN] <column name> { CASCADE | RESTRICT }
```

Alter SQL tables

Syntax | SQL (Setup) | Adding a column

```
CREATE TABLE IF NOT EXISTS dsc_table
(
  trainee_id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT
    ,
  trainee_first_name CHAR
    ,
  trainee_last_name CHAR
);

INSERT
    INTO
    dsc_table (
        trainee_first_name
        , trainee_last_name
    )
VALUES (
    "First name"
    , "Last Name"
)
```

Alter SQL tables

Syntax | SQL (Setup) | Adding a column

```
ALTER TABLE dsc_table ADD trainee_join_date DATE DEFAULT "2201-01-01";
```

Using SQL within R

Resources

- SQL book
- Differences between RDBMS

Thank you !