

# SQL DDL

---

# Overview of this video

---

A run through of SQL DDL (DDL = Data Definition Language)

# How to create a database

Words in SQL are not  
case-sensitive!

CREATE DATABASE **database**name; = CrEaTe DaTaBaSe **database**name;

Creates a database named “**database**”

CREATE DATABASE **CS\_Store**;

Creates the database for our running example

(The database is conceptually similar to a folder, will initially be empty, and is used for holding your tables and other things involved in the database)

# How to create a table

What kind of white-space you use in SQL does not matter

```
CREATE TABLE Table_name (  
    column1 datatype,  
    column2 datatype,  
    column3 datatype  
);
```

=

```
CREATE  
TABLE Table_name  
    (column1 datatype, column2  
    datatype,  
    column3 datatype );
```

Creates a table named Table\_name with 3 columns, named column1, column2 and column3, each with a datatype of datatype

As a reminder: its schema is Table\_name(column1,column2,column3)

# Datatypes

---

INT – integers

FLOAT – decimal numbers

CHAR(x) – x is an integer, fixed length string

- E.g. CHAR(5) will give you a string, such that each such string has length exactly 5
  - E.g. “House” would fit in a CHAR(5) field, but formally, “Car” would not. In practice some padding will be added like “Car ” if you entered “Car”

VARCHAR(x) – x is an integer, variable length string

- E.g. VARCHAR(5) will give you a string of length at most 5, so both “House” and “Car” would fit

DATE – dates

- Format YYYY-MM-DD, e.g. 1990-11-10 would be the 10<sup>th</sup> of November 1990

DATETIME – for time and dates

- Format YYYY-MM-DD HH:MI:SS, e.g. 1990-11-10 17:15:00 would be a quarter past 5pm the 10<sup>th</sup> of November 1990

XML – for XML files

Will talk abt. XML later in course

BLOB – binary files (e.g. programs)

Others for other precisions

# Create table example

---

```
CREATE TABLE Employees (  
    birthday DATE,  
    first_name VARCHAR(100),  
    family_name VARCHAR(100)  
);
```

**Employees**

birthday	first_name	family_name
1990-11-10	Anne	Smith
2000-02-05	David	Jones
...	...	...


- As a reminder: the schema is Employees(birthday,first\_name,family\_name)

# Unique

---

You can specify that you want some attribute (or set of attributes) to be **unique**

```
CREATE TABLE Employees (  
    birthday DATE,  
    first_name VARCHAR(100),  
    family_name VARCHAR(100),  
    CONSTRAINT UC_Employees UNIQUE(birthday,first_name)  
);
```



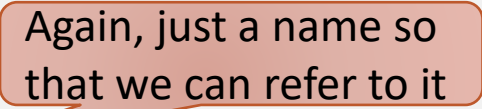
(To be explicit: Unique in a table means that for each value, there is at most one row in the table where the attribute/set of attributes take that value)

- Hence, we could have two employees with the same first name, but not with the same first name and birthday

# Primary Key

You can specify how the data should be sorted physically in storage by defining a **primary key**

```
CREATE TABLE Employees (  
    birthday DATE,  
    first_name VARCHAR(100),  
    family_name VARCHAR(100),  
    CONSTRAINT PK_Employees PRIMARY KEY (birthday,first_name)  
);
```



Primary keys must be unique (in the same sense as on the last slide) and there can only be 1 primary key per table (there can be many unique attributes/sets of attributes)

- Primary keys are often id number, e.g. student id/employee id or similar



# Foreign Key

---

A foreign key is used to link two tables together explicitly

- It is pointing from some attributes in one table, the child table, to the primary key of another, the parent table
- It ensured that the values in the attributes with the foreign key in the child table must also be in the parent table
  - Say that employee\_id is the primary key for the employee table and we have a table that contains transactions, with an associated employee\_id (i.e. for the employee that sold the items in the transaction to the customer). We can then specify that the employee\_id in the transaction table should have a foreign key to the employee\_id of the Employee table, ensuring that we get an error msg if we mistype the employee\_id in the transaction table (unless the mistype matches some other employee)
- Can be used to create **custom datatypes**, by having a table with all the values you want to allow
  - E.g. you could have table with all the post codes in the UK, letting it be the primary key and then using a foreign key to this table when you want people to specify post code

# Foreign Key cont.

```
CREATE TABLE Employees (  
  e_id INT,  
  first_name VARCHAR(100),  
  family_name VARCHAR(100),  
  CONSTRAINT PK_Employees PRIMARY KEY (e_id) );
```

```
CREATE TABLE Transactions (  
  CONSTRAINT FK_Transactions FOREIGN KEY (emp_id)  
  REFERENCES Employees(e_id) );
```

Name of parent table

Again, just a name so  
that we can refer to it

Name of attribute in child table

Attribute name of primary key

# How to remove things

---

Delete or remove is called DROP in SQL

Examples:

```
DROP DATABASE CS_Store;
```

Removes the `CS_Store` example database

```
DROP TABLE Employees;
```

Removes the `Employees` table

One can also remove unique, primary key or foreign keys similarly, but that is done extremely rarely, so I would suggest you look it up if you need it

# Modifying tables

---

```
ALTER TABLE Employees ADD email VARCHAR(100);
```

- Adds an email attribute

Exact command depends on implementation (this is the definition for MySQL and Oracle)

```
ALTER TABLE Employees MODIFY email VARCHAR(200);
```

- Changes the email attribute to allow longer emails

```
ALTER TABLE Employees DROP COLUMN email;
```

- Removes the email attribute again