**Day 1:**

**Database** - a structured set of data held in a computer, especially one that is accessible in various ways.

We must be able to structure tables to contain the right type of information, so that we can query it.

**Column** - Database tables are composed of individual columns corresponding to the attributes of the object.

**Row** - A row consists of one set of attributes corresponding to one instance that a table describes. Known as Records or Tuples.

**Table** - A predefined format of rows and columns that define an entity. Known as a File.

**DBMS** - A DataBase Management System allows a computer to perform database functions of storing, retrieving, adding, deleting and modifying data

Primary Key Constraints:

* Must be unique
* Must always have an entry (cannot be blank or Null)
* The value must never change
* Each table may have a maximum of one Primary Key
* Not a constraint, but Primary Key is known as a ‘simple’ type

**Compound/Composite Key** - Combines more than one field to make a unique value and is used as the primary key.

**Candiate Key** - Fields that are sufficient in being a Primary Key.

**Foreign Key** - A Primary Key from another table used to create solid relationships between two tables.

* Ensures that the row of information in Table A corresponds to the correct row of information in Table B.
* Constraint is used to prevent actions that would destroy links between tables.
* Prevents invalid data from being inserted into the foreign key column, it has to be a value the contained in the table it points to.
* No uniqueness constraints for them.
* A table can have any number of Foreign keys.
* A row cannot be deleted from a reference table if it is in use via foreign key.

**Flat-file Database** - Stores everything in one table. Good for small numbers of records related to a single topic.

**Relational Database** - Gives you the ability to separate masses of data into numerous tables. They are linked to each other through the use of keys.

**Big Data** - MongoDB, Vertica etc.

* Used for Data Analytics and Business Intelligence.
* Digital Age and Internet of Things

Relationship Types:

One-to-one:

One-to-many:

Many-to-many:

Good practice to turn it into one-to-many by using a Junction table, use two primary keys

as a composite key for that table

***Types of Structure Query Language (SQL):***

The operations in SQL can be divided into multiple categories.

* **Data Manipulation Language**: Handles the manipulation of data in the table and can be used to access, modify or delete existing records.
  + SELECT – retrieve data from a table
  + INSERT – insert data into a table
  + UPDATE – updates existing data within a table
  + DELETE – deletes records from a table
* **Data Definition Language:** Used to define the table schema as well as modifying the table’s structure.
  + CREATE – allows the creation of tables
  + ALTER – modifies of the structure of an existing table. E.g. add, delete, modify columns, and add constraints.
  + DROP – deletes the whole table and its structure.
  + TRUNCATE – empties a table’s records while keeping its structure intact.
* **Data Control Language**: handles user permissions on the database
  + GRANT – grant permissions to a user.
  + REVOKE – to revoke permissions.
* **Transaction Control Language:** Used to manage transactions within a database.
  + COMMIT – permanently saves a transaction into the database
  + ROLLBACK – reverts to a previous savepoint in case of errors.
  + SAVEPOINT – temporarily saves a transaction for rollback purposes.

***Data Types:***

**VARCHAR** - Adaptable to different lengths of characters. Records MAX size.

**CHARACTER** or **CHAR** - Data must be at a fixed length. Fixed amount of space used.

**INT** - Holds a while number/integer value (see also bigint, smallint and tinyint) positive or negative.

**DATE** or **TIME** or **DATETIME** - Stores Date, Time or both date and time.

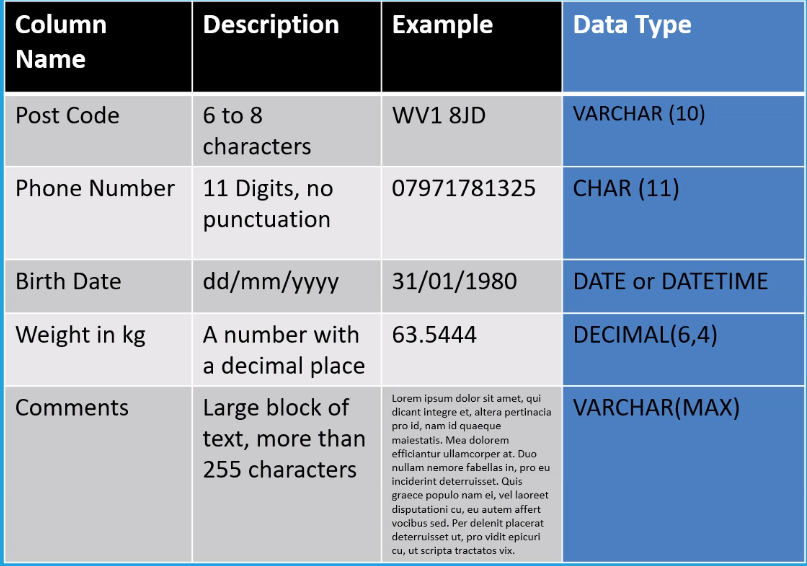
**DECMINAL** or **NUMERIC** - Fixed Precision and scale (digits to the right of the decimal point) numbers. E.g. 65.2345 has 6 precision, 4 scaling, or written as DECIMAL(6, 4).

**BINARY** - used to store binary data such as an image or file.

**FLAT** - Scientific use (very large numbers).

**BIT** - Equivalent to binary (0, 1, or NULL) (Boolean stuff etc.)

**DETETE On cascade** retains data integrity with deleting records. For instance, deleting the primary key entity will remove the foreign key entities from the other tables.



**Day 2:**

**Variations on INSERT:**

* Changing the order of the columns
* Omitting column names
* Leaving some columns out

VARCHAR, CHAR and DATE use single quotes.

DECIMAL and INT does not use any quotes.

**Null**

* NULL is not nothing but doesn’t equal zero.
* It’s not even an empty string.
* A value can be NULL, but NULL never equals to NULL because NULL is an undefined value.

By using NOT NULL, we can force the table not to accept empty fields for that column.

**Default**

Sets a default value for a field if one isn’t specified.

**UPDATE**

* Changes the contents of the table.
* Leaving the WHERE cause will update the whole table.

**DELETE**

* Removes the rows from a table based on the WHERE clause.
* Leaving the WHERE clause will empty the entire table.

**Database Considerations**

* Data Security
* Data Recovery
* Data Integrity
* Normal Form

***Normal(isation) Form***

Normalisation is a database design technique that reduces data redundancy and removes undesirable effects from operations like INSERT, UPADATE, and DELETE.

**First Normal Form:**

A database is in the First Normal Form when the following conditions are satisfied:

* All data values should be Atomic
* i.e. table entries should be single values, not sets or composite objects
* Simplifies queries and data comparisons

**Second Normal Form:**

* Is in 1NF
* All non-key attributes are fully functionally dependent on the primary key.

*Example:*

In this teacher table, the teacher’s age is dependent on just the TeacherID not the subject. This means the age is not fully dependent on the primary key (TeacherID, Subject). Split into teacher table and teacher subject table.

|  |  |  |
| --- | --- | --- |
| **TeacherID** | **Subject** | Age |
| 1 | Maths | 38 |
| 1 | Physics | 38 |
| 2 | Biology | 40 |
| 3 | Physics | 45 |
| 3 | Chemistry | 45 |

|  |  |
| --- | --- |
| **TeacherID** | Age |
| 1 | 38 |
| 2 | 40 |
| 3 | 45 |

|  |  |
| --- | --- |
| **TeacherID** | **Subject** |
| 1 | Maths |
| 1 | Physics |
| 2 | Biology |
| 3 | Physics |
| 3 | Chemistry |

**Third Normal Form:**

* Is in 2NF
* There is no transitive function dependency.
* Transitive dependency is a functional dependency between two (or more) non-key attributes.

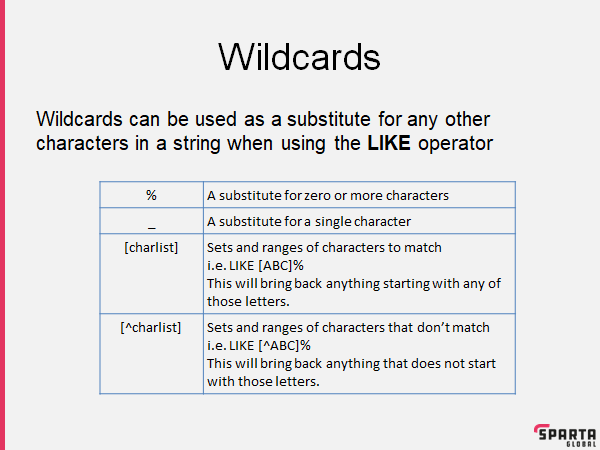
*Example:*

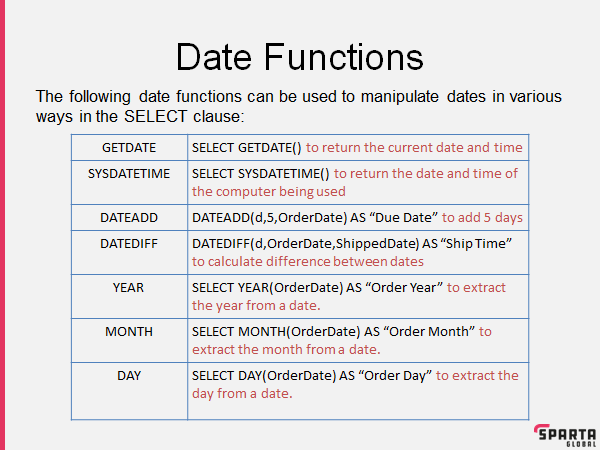
In the films table, description is dependent on the certificate (not the filmID), so the certificate and description will be split into its own table.

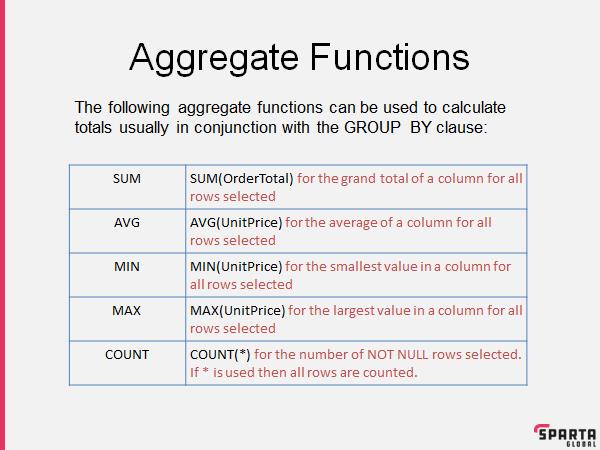
|  |  |  |  |
| --- | --- | --- | --- |
| **FilmID** | Title | Certificate | Description |
| 1 | Predator | 18 | Eighteen and over |
| 2 | Monster House | PG | Parental Guidance |
| 3 | The Lorax | U | Universal Classification |

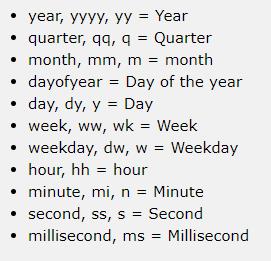
|  |  |
| --- | --- |
| **Certificate** | Description |
| 18 | Eighteen and over |
| PG | Parental Guidance |
| U | Universal Classification |

|  |  |  |
| --- | --- | --- |
| **FilmID** | Title | Certificate |
| 1 | Predator | 18 |
| 2 | Monster House | PG |
| 3 | The Lorax | U |





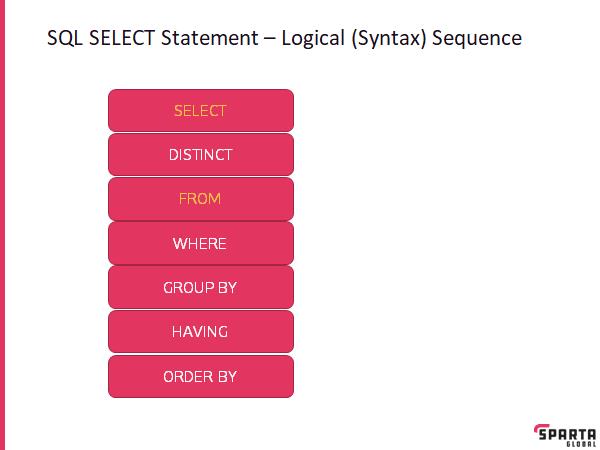
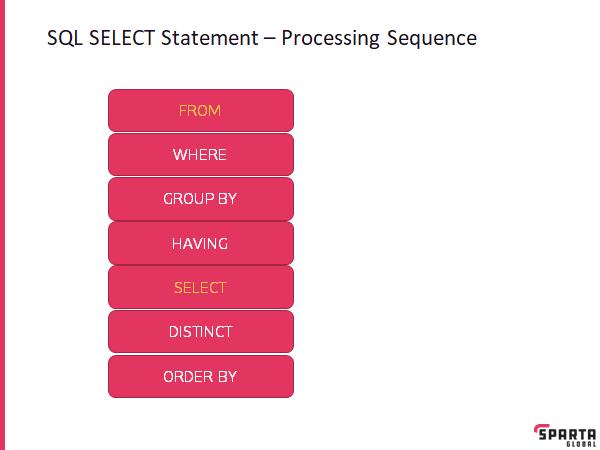


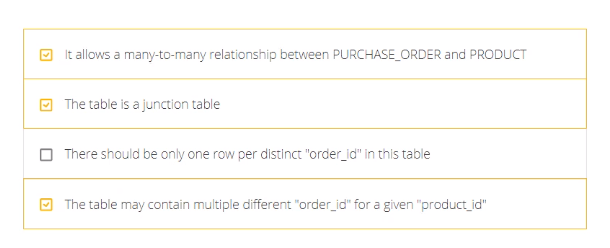


**Day 3:**

**JOIN** is used to combine tables with matching fields.

* **INNER:** Only takes Key matching rows will be included. Everything else is discarded.
* **LEFT:** Everything on the left table matches with items on the right table. Any left values unmatched will be NULL.
* **RIGHT:** Everything on the right table that matches with the left table will show. Any right values unmatched will be NULL.
* **FULL:** Everything with the left and right are joined, matching values are shown. Anything not matching will have NULL values.

****

**Junction Tables:**